

04/15/99
JC645 U.S. PTO

PATENT
Attorney Docket No.: ORAS1100-1

APPLICATION FOR U.S. PATENT
TRANSMITTAL FORM

THE COMMISSIONER OF PATENTS
AND TRADEMARKS
Washington, D.C. 20231

JC618 U.S. PTO

09/293536

04/15/99

Sir:

Transmitted herewith for filing is the patent application of:

In the Application of: Alexander G. Parlos, et al.
Date Filed: April 15, 1999
Title: SYSTEM AND METHOD FOR CONDITION ASSESSMENT AND
END-OF-LIFE PREDICTION
Enclosed are: 2 Sheets of Informal Drawings

FEE CALCULATION					FEE
	Number of Claims	Number of Allowed Claims	Number Extra	RATE	BASIC FEE
Total Claims	1	-20	<19>	x \$9 =	\$ 380.00
Independent Claims	1	- 3	<2>	x \$39 =	\$
TOTAL FILING FEE =					\$ 380.00

The Commissioner is hereby authorized to deduct \$380.00 from Deposit Account No. 50-0456, and to charge any additional fees or credit any overpayment to Deposit Account No. 50-0456 of Gray Cary Ware & Freidenrich LLP.

GRAY CARY WARE & FREIDENRICH LLP

Steven R. Sprinkle
Registration Number: 40,825

Dated: April 15, 1999
100 Congress Avenue, Suite 1440
Austin, Texas 78701
(512) 457-7025
(512) 457-7070 (facsimile)

Attorney Docket No.: ORAS1100-1
Applicant or Patentee: Alexander G. Parlos, et al.
Filed: Herewith
Title: SYSTEM AND METHOD FOR CONDITION ASSESSMENT AND END-OF-LIFE PREDICTION

**VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS
(37 C.F.R. §§ 1.9(f) and 1.27(c)) - SMALL BUSINESS CONCERN**

I hereby declare that I am

- ☐ the owner of the small business concern identified below:
☒ an official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF CONCERN: Orasis Software, Inc.
ADDRESS OF CONCERN: 3355 Bee Caves Road, Suite 203
Austin, Texas 78746

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 C.F.R. § 121.3-13 and reproduced in 37 C.F.R. § 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention, entitled SYSTEM AND METHOD FOR CONDITION ASSESSMENT AND END-OF-LIFE PREDICTION by inventors Alexander G. Parlos, Omar T. Rais, Sunil K. Menon, Armit F. Atiya, Hamid A. Toliyat, Esmaeil Oufi, Alton D. Patton and Benito Fernandez, described in:

- ☒ the specification filed herewith with title as listed above.
☐ the application identified above.
☐ the patent identified above.

If the rights held by the above identified small business concern are not exclusive, each individual, concern or organization having rights to the invention is listed below and no rights to the invention are held by any person, other than the inventor, who could not qualify as a small business concern under 37 C.F.R. § 1.9(d) or by any concern which would not qualify as a small business concern under 37 C.F.R. § 1.9(d) or a nonprofit organization under 37 C.F.R. § 1.9(e).

NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities (37 C.F.R. § 1.27).

Full Name _____
Address _____
☐ Individual ☒ Small Business Concern ☐ Nonprofit Organization

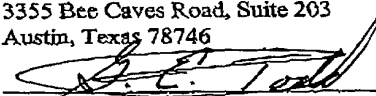
I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate (37 C.F.R. 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING
TITLE OF PERSON OTHER THAN OWNER
ADDRESS OF PERSON SIGNING

George E. Todd
Vice President Finance
3355 Bee Caves Road, Suite 203
Austin, Texas 78746

SIGNATURE



DATE

April 15, 1999

SYSTEM AND METHOD FOR CONDITION
ASSESSMENT AND END-OF-LIFE PREDICTION

5 RELATED APPLICATION

 This application claims priority under 35 U.S.C.
§ 119(e)(1) to provisional application number 60/081,848
filed April 5, 1998.

10 TECHNICAL FIELD OF THE INVENTION

 The present invention relates generally to systems
and method for providing just-in-time maintenance for
equipment, and more particularly, to a system and method
15 for providing an assessment of the condition of a piece
of equipment or an entire system (i.e., whether
maintenance is required) and for providing a prediction
for the equipment/system end-of-life.

20

BACKGROUND OF THE INVENTION

In manufacturing, power generation, oil & gas production and refining, and milling sectors, the failure of critical components results in lost revenues and emergency maintenance costs. Industry's response to this risk has been to invest heavily in scheduled preventive maintenance. The importance of detecting problems and preventing failures is reflected in the fact that as much as 15% to 40% of manufacturing production cost is allocated to maintenance. Maintenance cost is one of the highest controllable operation costs. A reliable proactive predictor of maintenance requirements of critical equipment would result in industry savings from reduced lost revenues, overtime costs associated with emergency repairs, and disrupted production schedules.

Current Just-in-Time (JIT) maintenance methods have attempted to address these issues. JIT maintenance means taking a piece of equipment off-line for servicing when it needs it, rather than according to a fixed schedule. It is expensive and time consuming to shut down critical equipment like motors, pumps, compressors and generators for maintenance, so plant operators would like to be sure that the equipment needs servicing before they schedule it. Today, maintenance schedules are based on manufacturer's specification test data. Fixed maintenance schedules result in shutting down a piece of equipment before it really needs it, or in continuing to

operate one that should be overhauled. They do not take in account equipment operating history, loading profiles, and operating environments. These are some of the key factors that determine equipment life expectancy.

5 Current technologies typically do not measure the long-term performance and assess the health of equipment while the equipment is operating. Nor is it possible to predict equipment failures well in advance of their occurrence for adequate planning. Experience indicates
10 that most often equipment fails when least expected and quite often immediately after a major overhaul. The life-time benefit that can be derived by a technology capable of assessing and predicting the long-term health of equipment is quite significant.

SUMMARY OF THE INVENTION

The present invention provides a system and method for condition assessment and end-of-life prediction that substantially eliminates or reduces disadvantages and problems associated with previously developed equipment maintenance systems and methods.

According to one aspect of the present invention, the condition assessment and end-of-life prediction system of the present invention includes two virtual instruments: a virtual condition assessment instrument and a virtual end-of-life prediction instrument. The virtual condition assessment instrument measures the condition of the equipment and includes a data capture subsystem for sampling a set of analog signals and converting them into digital signals, a model-based component to estimate disturbances and predict an expected response, a signal-based component to process output from the model-based component, a classification component to process output from the signal-based component, a fuzzy logic expert component to combine information from the classification component and the model-based component in order to assess the condition of the equipment, and a condition assessment panel to display the condition of the equipment. The a virtual end-of-life prediction instrument predicts the equipment end-of-life and includes a condition prediction end-of-life prediction component to analyze information from the

virtual condition assessment instrument to predict
condition and end-of life, a prediction condition and
end-of-life uncertainty estimation component to estimate
the uncertainty of the condition and end-of-life
5 prediction, and
an end-of-life panel for displaying the condition and
end-of-life prediction and uncertainty.

A technical advantage of the present invention is
the use of software programming that uses historical data
10 to indicate when a piece of equipment is out of
calibration or in need of service. This technical
advantage allows the user of the equipment to minimize
down time by eliminating fixed schedule off-line
servicing. This eliminates both shutting down a piece of
15 equipment before it really needs it and continuing to
operate one that should be overhauled.

Another technical advantage of the present invention
is the use of software programming that uses historical
data to predict the end-of-life of a piece of equipment.
20 The present invention measures the long-term performance
and assesses the health of equipment during operation.
This allows a user to (1) predict equipment failures well
in advance of their occurrence and (2) only replace
equipment that is actually approaching end of life.

25 The present invention provides yet another technical
advantage by providing a reliable proactive predictor of
maintenance requirements of critical equipment that

results in cost savings due to a reduction in equipment down time, overtime costs associated with emergency repairs, and disrupted production schedules.

AU\4013828.2
103648-990000

DETAILED DESCRIPTION

Implementation of the condition assessment and end-of-life prediction maintenance technology of the present invention is based on the following technological
5 innovations:

- Signal processing algorithms and software programs for: (1) multi-step-ahead (including single-step-ahead) predictor (or forecasting) systems in data-rich and data-scarce
10 environments, (ii) nonlinear disturbance estimators, (iii) nonlinear state filters, and, (iv) the uncertainty associated with the estimates in (i), (ii) and (iii),
- Intelligent Condition Assessment and End-of-Life Prediction System (ICAPS) utilizing physical (or
15 hardware) instruments (or sensors) as inputs and inferring the system condition and system end-of-life (or remaining useful life or residual life) as outputs, including the uncertainty associated with these inferences,
20
- Virtual (or software) instruments (or sensors) displaying equipment condition and equipment end-of-life information.

FIGURE 1 shows an overview of the invention in broad
25 detail and the interrelation of the various parts of the invention are presented.

ENABLING SIGNAL PROCESSING TECHNOLOGY

The signal processing technology at the core of the
JIT maintenance technology has been developed over the
last ten years.

Neural network software is at the heart of our
information processing technology.— Neural networks are
one of the most promising mechanisms to supply reliable
and critical timely information. Our neural network's
unique ability to learn the characteristics of man-made
dynamic systems comes from the introduction of feedback
into a conventional feed-forward architecture.

The signal processing developments deal with
estimation in nonlinear systems, in general. Algorithms
that enable the construction of nonlinear predictors, in
general, have been developed. These predictors are
appropriate for multi-step-ahead prediction, in general,
including single-step-ahead prediction in data-rich
environments. The construction methods are applicable to
non-adaptive and adaptive predictors. The architectures
(or model structures) that this invention can apply to
include, but are not limited to, the one presented in US
Pat. No. 5,479,571, hereby incorporated by reference in
its entirety. TAMUS 1058 presents one embodiment of this
invention incorporated into the architecture of US Pat.
No. 5,479,571, hereby incorporated by reference in its
entirety.

Additionally, algorithms that enable the construction of nonlinear state filters, in general, have been developed. Methods have been developed for the construction of non-adaptive, adaptive and hybrid state filters in data-rich environments, as described in detail later. The architectures (or model structures) that this invention applied to includes, but is not limited to, the one presented in US Pat. No. 5,479,571. TAMUS 1084 presents one embodiment of this invention incorporated into the architecture of US Pat. No. 5,479,571, hereby incorporated by reference in its entirety.

The last component of the enabling signal processing technology consists of algorithms for the multi-step-ahead prediction (or forecasting) in data-scarce environments. Because the associated uncertainty in data-scarce environments is large, a forecast uncertainty estimation algorithm has also been developed. The architectures (or model structures) that this invention applies to includes, but is not limited to, the one presented US Pat. No. 5,479,571, hereby incorporated by reference in its entirety. TAMUS 1097 presents one embodiment of this invention incorporated into a special form of the architecture in US Pat. No. 5,479,571, hereby incorporated by reference in its entirety.

INTELLIGENT CONDITION ASSESSMENT AND END-OF-LIFE
PREDICTION SYSTEM (ICAPS)

The Intelligent Condition Assessment and End-of-Life Prediction System (ICAPS) consists of a series of signal processing algorithms combined in unique ways to allow:

(i) assessment of equipment condition and the associated uncertainty, and (ii) prediction of equipment end-of-life and the associated uncertainty. FIGURE 2 shows a system-level description of the Intelligent Condition Assessment and End-of-Life Prediction System. Figure 2 depicts the ICAPS as receiving inputs from the equipment physical sensors and the signal processing algorithms. The ICAPS can be implemented using signal processing algorithms other than the ones presented in this document. The present embodiment of ICAPS in this document depends on the signal processing technology of TAMUS 1058, TAMUS 1084 and TAMUS 1097, as shown in FIGURE 1.

A more detailed description of the operation and implementation of the Intelligent Condition Assessment and End-of-Life Prediction System (ICAPS) is provided A below.

VIRTUAL INSTRUMENTS (OR SENSORS) FOR MEASURING EQUIPMENT
CONDITION AND EQUIPMENT END-OF-LIFE

This section relates to the virtual (software) instrument (or sensors) for measuring the long-term equipment condition and equipment end-of-life aspects of

the invention. There are no physical (or hardware) sensors that can measure equipment condition or end-of-life directly. Therefore equipment condition and end-of-life measurements must be inferred by other direct (or physical) measurements and by the use of virtual sensors, as shown in Figure 3.

A more detailed description of the operation and implementation of the Virtual Instruments is provided below.

Virtual Condition Instrument

A virtual equipment condition instrument is defined to be a software system that is connected to a physical piece of equipment through physical (or hardware) sensors and which can accurately, continuously, non-intrusively, and in real-time or in near real-time provide equipment condition information, i.e. provide equipment condition information without the need to disrupt equipment operation and without human intervention. Here condition is broadly defined to reflect (a) the current status of incipient failures and the associated uncertainties, (b) the repairs appropriate for the current status and the costs associated with the (i) direct labor, (ii) parts, and (iii) down-time to accomplish these repairs, (c) the equipment efficiency and the costs associated with the efficiency degradation.

A more detailed description of the operation and implementation of the Virtual Condition Instrument is provided below.

5 Virtual End-of-Life Instrument

 A virtual equipment end-of-life instrument is defined to be a software system that is connected to a physical piece of equipment through physical (or hardware) sensors and which can accurately, continuously, non-intrusively, and in real-time or in near real-time provide equipment end-of-life information, i.e. provide equipment end-of-life information without the need to disrupt equipment operation and without human intervention. Here end-of-life (or remaining useful life or residual life) is broadly defined to reflect (a) expected time to failure and the associated uncertainty, (b) the predicted status of incipient failures, (c) the repairs appropriate for the predicted status and the costs that will be associated with the (i) direct labor, (ii) parts, and (iii) down-time to accomplish these predicted repairs, (d) the predicted equipment efficiency and the costs associated with the predicted efficiency degradation.

 A more detailed description of the operation and implementation of the Virtual End-Of-Life Instrument is provided below.

The present invention can include the following features:

1. SIGNAL PROCESSING

- [1] Adaptive single-step-ahead prediction of measured
5 complex system output variables, where the complex
system comprises of nonlinear, stochastic and
generally unknown dynamics.
- [2] Nonadaptive filtering of unmeasurable (or
unmeasured) complex system state variables, where
10 the complex system comprises of nonlinear,
stochastic and generally unknown dynamics.
- [3] Adaptive filtering of unmeasurable (or unmeasured)
complex system state variables, where the complex
15 system comprises nonlinear, stochastic and generally
unknown dynamics.
- [4] Hybrid (nonadaptive and adaptive) filtering of
unmeasurable (or unmeasured) complex system state
20 variables, where the complex system comprises of
nonlinear, stochastic and generally unknown
dynamics.
- [5] Adaptive multi-step-ahead prediction (forecasting)
of measured complex system output variables, where
the complex system comprises of nonlinear,
25 stochastic and generally unknown dynamics.
- [6] Uncertainty estimation (confidence interval
computation) of an adaptive multi-step-ahead
predictor (forecasting system) of measured complex

system output variables, where the complex system comprises of nonlinear, stochastic and generally unknown dynamics.

5 2. SYSTEM LEVEL DIAGNOSIS AND PROGNOSIS

-- [1] Model-based diagnosis of incipient failures in complex systems, where the complex system comprises of nonlinear, stochastic and generally unknown dynamics.

10 [2] Signal-based diagnosis of incipient failures in complex systems, where the complex system comprises of nonlinear, stochastic and generally unknown dynamics

15 [3] Hybrid- signal-based and model-based - diagnosis or incipient failures in complex systems, where the complex system comprises of nonlinear, stochastic and generally unknown dynamics.

20 [4] Decoupling the effects of system inputs and disturbances on the system outputs, from the effects of system incipient faults.

[5] Prognosis of incipient failures and prediction of the end-of-life of complex systems, where the complex system comprises of nonlinear, stochastic and generally unknown dynamics.

25 [6] Estimating the uncertainty in the end-of-life of complex systems, where the complex system comprises

of nonlinear, stochastic and generally unknown
dynamics.

3. SYSTEM SPECIFIC DIAGNOSIS AND PROGNOSIS

- 5 [1] Model-based diagnosis of incipient failures in
electric motors, electric generators of all types
(electric machines, in general), electric
transformers of all types, electric batteries of all
types, electric motor-driven equipment of all types,
10 i.e. pumps, fans, compressors, machine tools,
valves, conveyor belts, prime movers of all types,
i.e. turbomachinery, diesel engines, internal
combustion engines, and process equipment, i.e.
boilers, heat exchangers.
- 15 [2] Hybrid - signal-based and model-based - diagnosis of
incipient failures in electric motors, electric
generators of all types (electric machines, in
general), electric transformers of all types,
electric batteries of all types, electric motor-
20 driven equipment of all types, i.e. pumps, fans,
compressors, machine tools, valves, conveyor belts,
prime movers of all types, i.e. turbomachinery,
diesel engines, internal combustion engines, and
process equipment, i.e. boilers, heat exchangers.
- 25 [3] Canceling the effects of poor electric power quality
from the motor stator current, electric generators
of all types (electric machines, in general),

electric transformers of all types, electric
batteries of all types, electric motor-driven
equipment of all types, i.e. pumps, fans,
compressors, machine tools, valves, conveyor belts,
5 prime movers of all types, i.e. turbomachinery,
diesel engines, internal combustion engines, and
process equipment, i.e. boilers, heat exchangers.

[4] Canceling the effects of load torque variations from
the motor stator current, electric generators of all
10 types (electric machines, in general), electric
transformers of all types, electric batteries of all
types, electric motor-driven equipment of all types,
i.e. pumps, fans, compressors, machine tools,
valves, conveyor belts, prime movers of all types,
15 i.e. turbomachinery, diesel engines, internal
combustion engines, and process equipment, i.e.
boilers, heat exchangers.

[5] Prognosis of incipient failures and prediction of
the end-of-life of electric motors, electric
20 generators of all types (electric machines, in
general), electric transformers of all types,
electric batteries of all types, electric motor-
driven equipment of all types, i.e. pumps, fans,
compressors, machine tools, valves, conveyor belts,
25 prime movers of all types, i.e. turbomachinery,
diesel engines, internal combustion engines, and
process equipment, i.e. boilers, heat exchangers.

[6] Estimating the uncertainty in the end-of-life of electric motors, electric generators of all types (electric machines, in general), electric transformers of all types, electric batteries of all types, electric motor-driven equipment of all types, i.e. pumps, fans, compressors, machine tools, valves, conveyor belts, prime movers of all types, i.e. turbomachinery, diesel engines, internal combustion engines, and process equipment, i.e. boilers, heat exchangers.

4. VIRTUAL INSTRUMENTATION

[1] A virtual instrument or sensor for measuring the condition (or health) of any type of equipment (all equipment categories listed in Section 3) in real-time.

[2] A virtual instrument or sensor for measuring the end-of-life of any type of equipment (all equipment categories listed in Section 3) in real-time.

Chapter 5 INTELLIGENT CONDITION ASSESSMENT AND END-OF-LIFE PREDICTION - ICAPS

In this chapter the Intelligent Condition Assessment and End-Of-Life Prediction System (ICAPS) architecture and its components are described in detail. The description is generic for any system with measured inputs and outputs, and it is not targeted for any specific equipment. Following this presentation, a specific implementation of the ICAPS components to induction motors is also described. After a system-level architectural description of ICAPS, the following ICAPS architecture components are further described:

- Condition assessment subsystem, and,
- End-of-life prediction subsystem.

The invention contained in this chapter was first disclosed in TAMUS 1059.

Section 1. ICAPS SYSTEM LEVEL DESCRIPTION

In this section we provide a “system-level” description of the Intelligent Condition Assessment and End-of-Life Prediction System (ICAPS) component of the JIT maintenance technology.

The underlying system architecture of the JIT maintenance technology is called the ICAPS, and it enables the implementation of the JIT maintenance technology. The ICAPS architecture performs two separate functions that are part of the functionality provided by the JIT maintenance system.

Figures 5-1, Figure 5-2 and Figure 5-3 depict the “system level” architectures for three different implementations of ICAPS. Figure 5-1 implements a “model-based” version of the ICAPS, where the outputs of the model-based subsystem are further processed by a signal-based subsystem in order to reduce the effects of the environmental and operating conditions thus reducing false alarms and missed faults. Figure 5-2 depicts a purely “signal-based” implementation of the ICAPS, where the role of the model-based subsystem is assumed by the fuzzy rule-based system, in addition to its role in deciding system condition. Thus the reduction of false alarms and missed faults is accomplished via a rule-based

system. Figure 5-3 implements a “hybrid” version of the ICAPS, consisting of a model-based and a signal-based subsystems operating in parallel. The functionalities provided by ICAPS are as follows:

- System Condition Assessment (Fault Diagnosis),
- System End-of-Life Prediction (Fault Prognosis).

Additionally, the ICAPS enables the application of the aforementioned two functions of the JIT maintenance technology, to system components that are mechanically, or otherwise, coupled, for example through a gear-box or by direct coupling.

The “system level” architecture of Figure 5-1 depicts the information flow from the sensor readings to the panels depicting the diagnosis and prognosis. The sensor signals are first conditioned and then sampled using a computer interface card. The sampled time-series are then processed through the “model-based” subsystem for decoupling two effects:

- The effects of the measured inputs on the measured outputs,
- The effects of the measured (or estimated) disturbances on the measured outputs.

Additionally, the “model-based” subsystem enables the down-stream system to detect novelty, that is to classify faults not previously seen by the pre-trained classifier.

The outputs of the “model-based” subsystem are the “output residuals”, that is the difference between the expected and observed system response. These residuals are then processed by the “signal-based” subsystem for extraction of any spatio-temporal features. This is accomplished by first transforming the signals to the time-frequency domain via one of possible transforms, such as simple FFT, windowed-FFT, or wavelet. Then the signals are processed through a multi-stage classifier for fault classification. The outputs from the classifier and the “model-based” subsystem are processed through a fuzzy (rule-based) expert system which assesses the system condition and reports it to the condition panel. Simultaneously, the condition information along with the sensed signals are processed through the end-of-life prediction system. This subsystem combines the current condition of the system with indicators, such as the ambient temperature, the electric power quality, the load cycling, and the number of start-ups to estimate the system remaining life and the uncertainty of the remaining life associated

with a given confidence level. In considering the aforementioned indicators, the integral of these indicators over time must be accounted for in the end-of-life prediction subsystem because of the cumulative effect they have on the life of the equipment. These results are reported to the end-of-life prediction system panel.

The “system level” architecture of Figure 5-2 depicts the information flow from the sensor readings to the panels depicting the diagnosis and prognosis. The sensor signals are first conditioned and then sampled using a computer interface card. The sampled time-series are then processed through the “signal-based” subsystem. The remaining processing is similar to the case shown in Figure 5-1.

The “system level” architecture of Figure 5-3 depicts the information flow from the sensor readings to the panels depicting the diagnosis and prognosis. The sensor signals are first conditioned and then sampled using a computer interface card. The sampled time-series are then processed in parallel through the “model-based” subsystem and the “signal-based” subsystem. The remaining processing is similar to the case shown in Figure 5-1.

One embodiment of the ICAPS implementations depicted in Figures 5-1, 5-2 and 5-3 is through the signal processing inventions described in TAMUS 1058, TAMUS 1084 and TAMUS 1097. However, this is not the only possible embodiment.

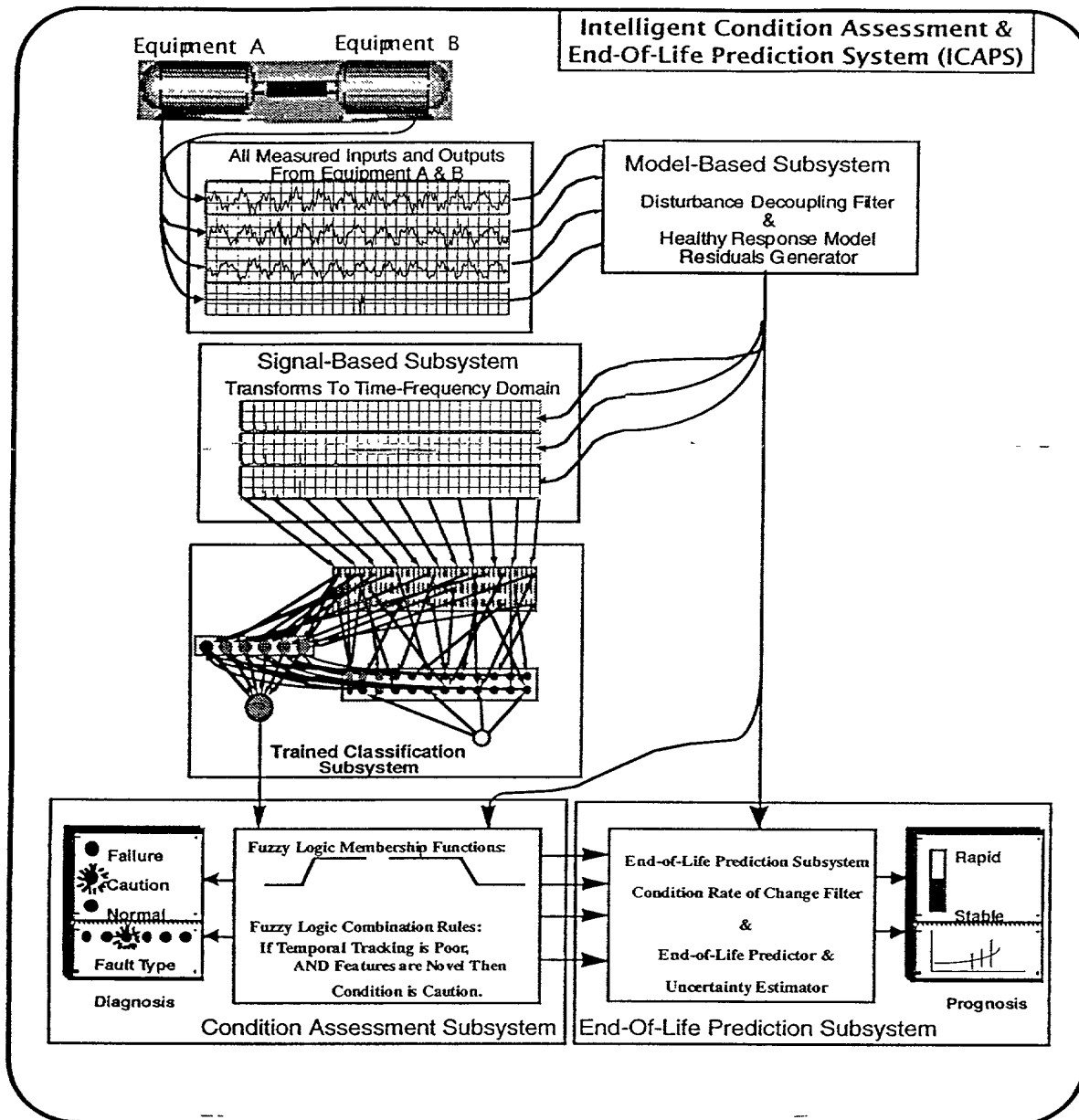


Figure 5-1 System-Level Description of JIT Maintenance Technology (Model-Based).

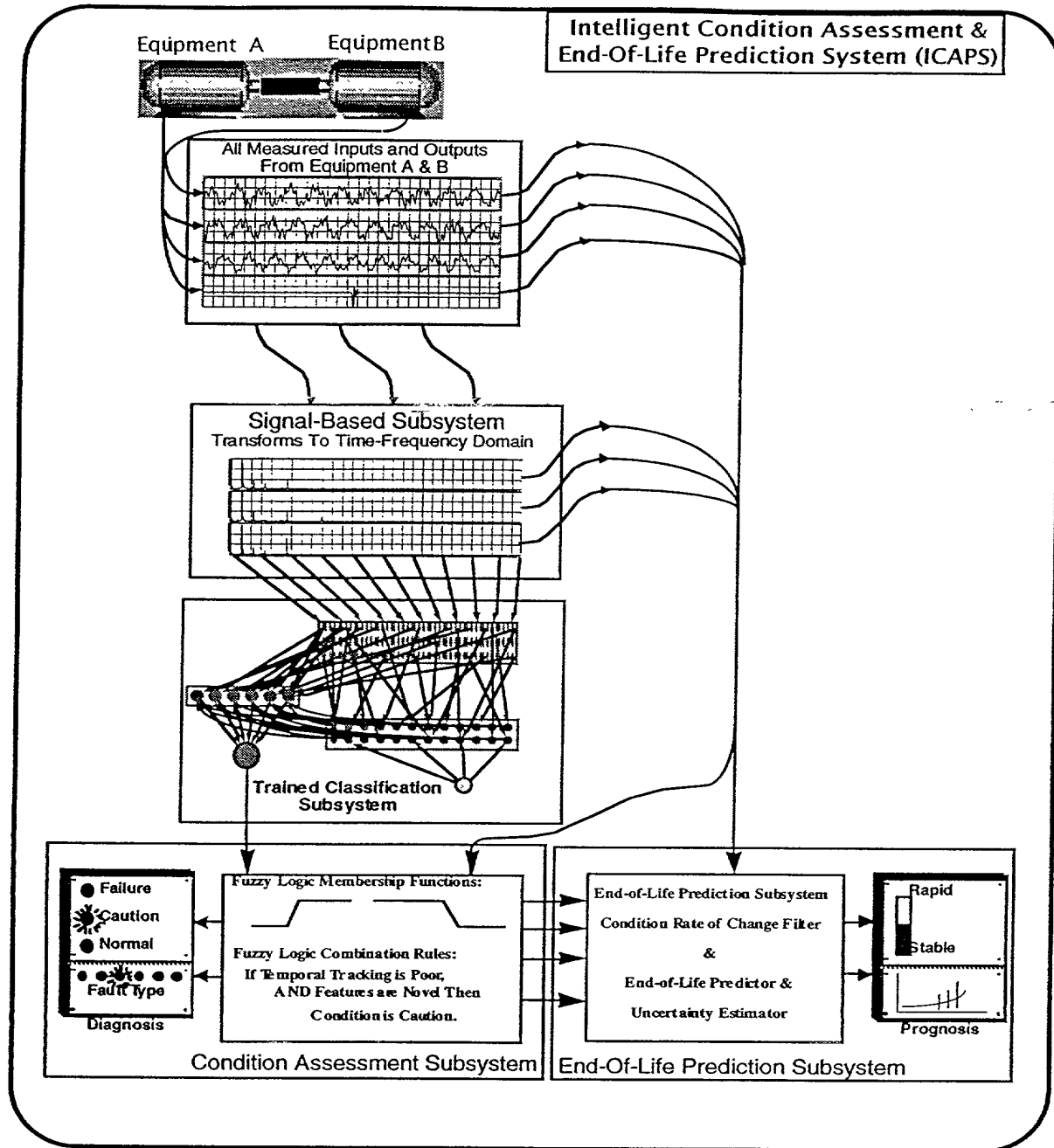


Figure 5-2 System-Level Description of JIT Maintenance Technology (Signal-Based).

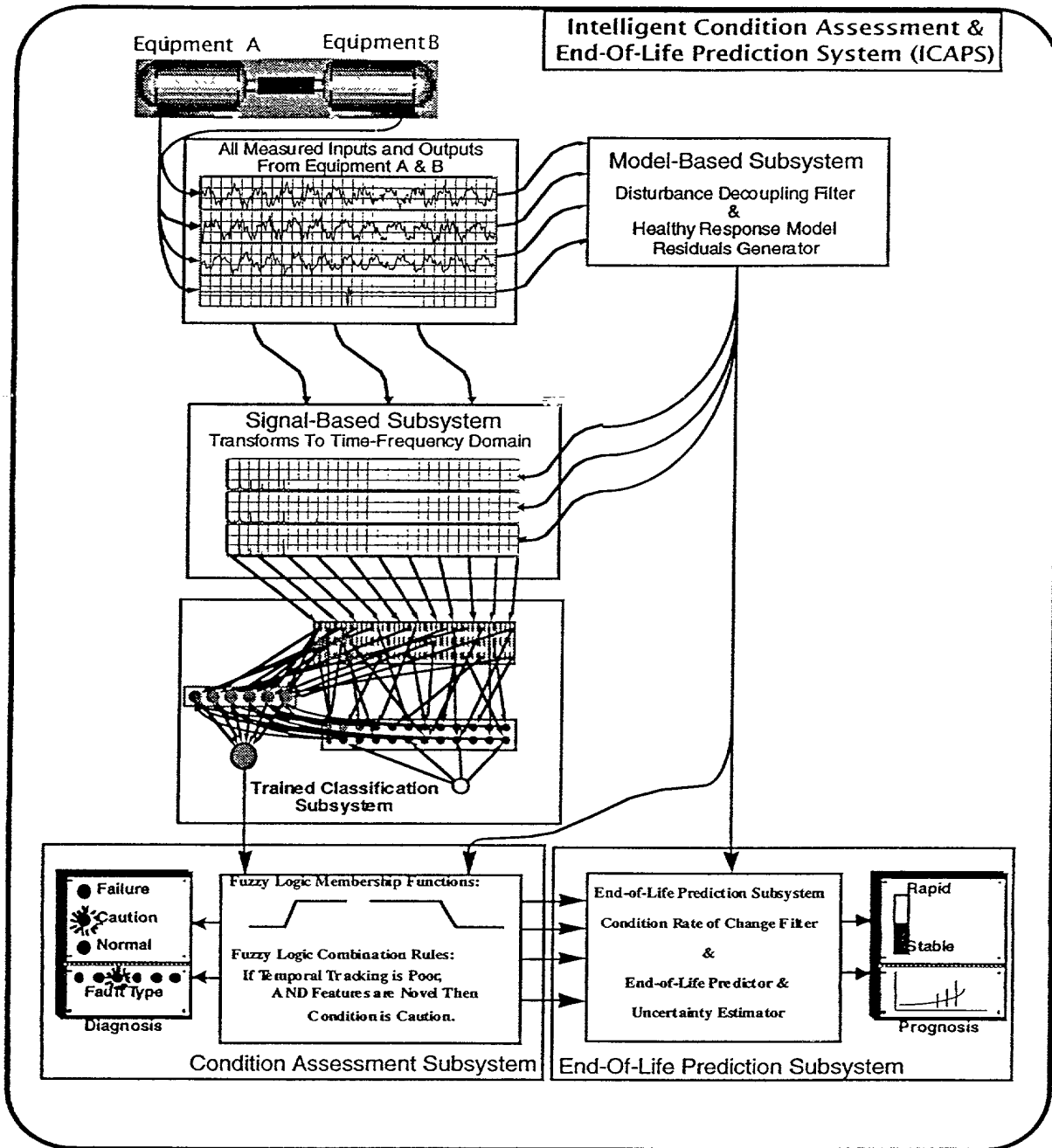


Figure 5-3 System-Level Description of JIT Maintenance Technology (Hybrid System).

Section 2. CONDITION ASSESSMENT SUBSYSTEM

The condition assessment subsystem is composed of a number of components which work as a whole to diagnose the current condition of the equipment being assessed. The specific components that make-up the condition assessment system are now described in more detail.

Subsection 1. Data Capture and Preprocessing Component

The Data Capture and Preprocessing Subsystem, depicted in Figure 5-1, is the interface between the analog signals of the real world and the digital signals of the computer (“virtual”) world. This subsystem is implemented in hardware, for example using DAQ cards, and it converts the many analog signals, both system inputs and outputs, it receives into digital signals via sampling. Prior to sampling the analog signals could be conditioned using analog filtering techniques to remove high frequency components or specific low frequency components. The sampling rate can be varying ranging up to 100 kHz or even 1 MHz.

Prior to sampling the signals are conditioned in analog form via the use of anti-aliasing low-pass filters, or other means of conditioning. Following sampling, the signals are conditioned and they are prepared for processing by the rest of the subsystems. The digital signal conditioning might take the form of additional digital filtering or scaling using some a priori known formulas. Depending on the system being diagnosed, the sampled signals could be electric voltage, electric current, rotational speed, rotational acceleration, lateral acceleration, temperature, etc.

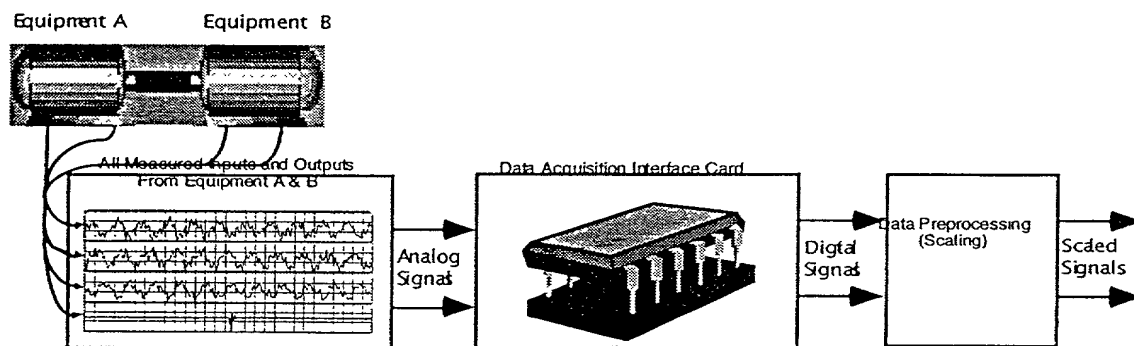


Figure 5-1 Data Capture and Preprocessing Component Block Diagram.

Subsection 2. Model-Based Component

The model-based component, depicted in Figure 5-2, serves two broad purposes: (1) to decouple the effects of the measured (or estimated) system inputs and disturbances on the system outputs, and (2) to enable novelty detection.

This component consists of two subcomponents: (1) a filter to estimate any disturbances that cannot be directly measured, for example to estimate the speed or torque of the load, (2) a predictor for predicting the expected (or “healthy”) response of the system under consideration. Both the filter and the predictor could be based: (1) on first-principles only, with some parameter tuning, (2) on purely empirical (or adaptively obtained) models, or (3) hybrid combination of the above two options. The filter

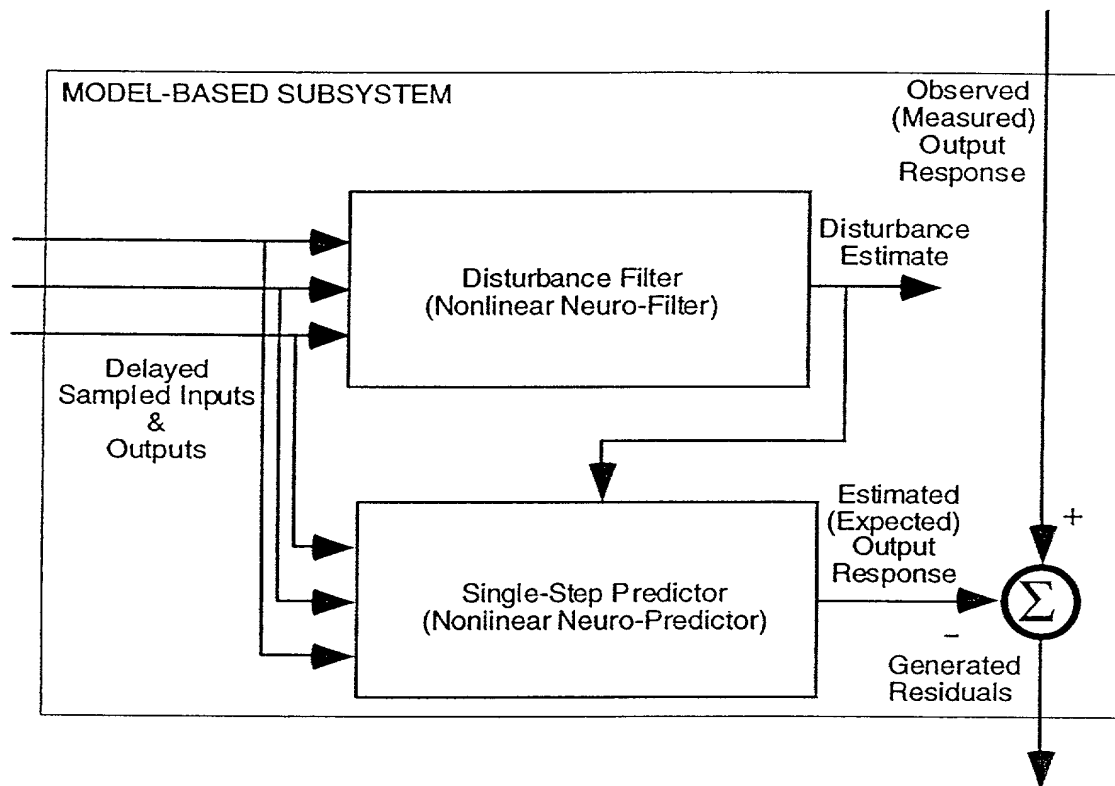


Figure 5-2 Model-Based Component Block Diagram.

of the model-based component can be implemented using the signal processing inventions of TAMUS

1084, and the predictor of the model-based component can be implemented using the signal processing inventions of TAMUS 1058. Implementations using other signal processing algorithms are also feasible.

Subsection 3. Signal-Based Component

The signal-based component, depicted in Figure 5-3, processes information from the model-based component outputs. This component system could also process information, in parallel, from the data capture and preprocessing component should that be necessary.

The signal-based component extracts spatio-temporal features related to faults. The expected signatures are canceled by the model-based component, therefore the signatures (features) extracted by the signal-based component belong to some fault class.

The signal-based component can be implemented using any of the available algorithms in the literature. The use of the signal-based component in combination with the model-based component, as done in ICAPS, and the use of the signal-based component in combination of the rule-based system, as done in ICAPS, gives a unique implementation.

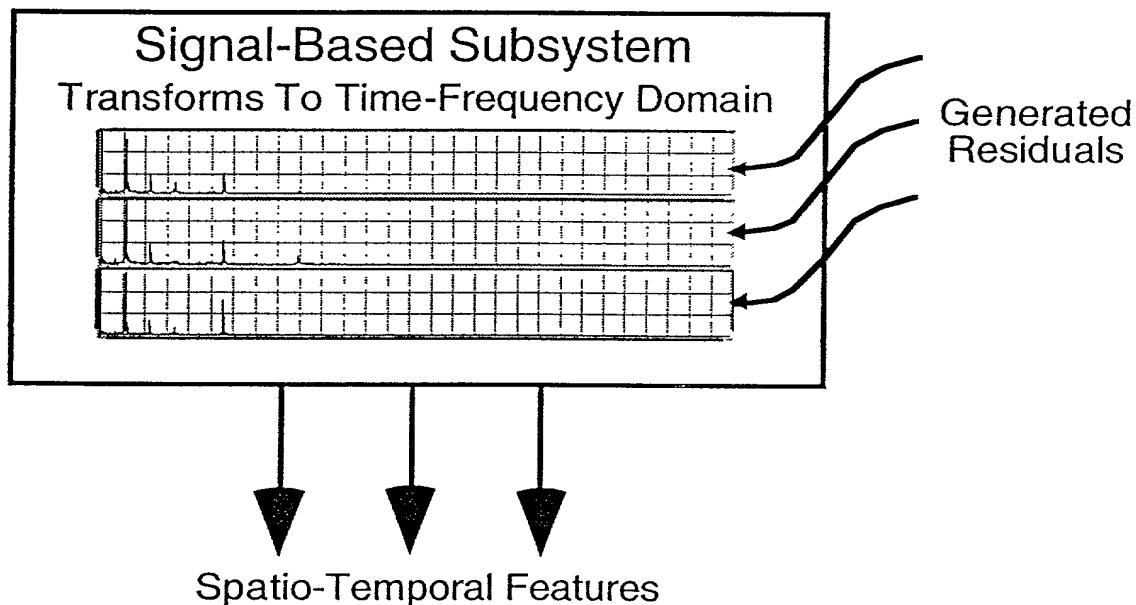


Figure 5-3 Signal-Based Component Block Diagram.

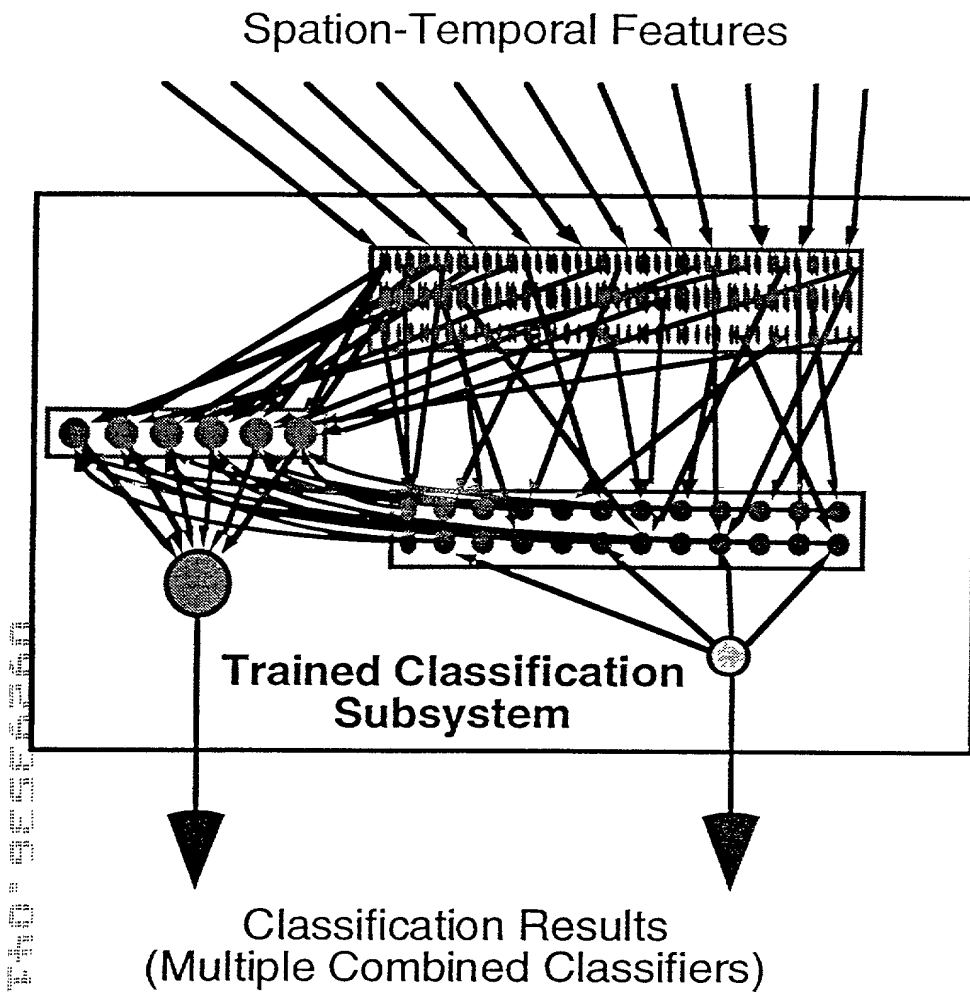


Figure 5-4 Classification Component Block Diagram.

Subsection 4. Classification Component

The classification component, depicted in Figure 5-4, processes information received from the signal-based component. This component is used to classify known failure modes. Having decoupled the features of “healthy” systems, the accuracy of the classification component is increased.

The classification component can be implemented using any of the available algorithms in the literature. The use of the model-based component outputs as inputs to the classification component, as done in ICAPS, gives a unique implementation.

Subsection 5. Fuzzy Logic Expert Component

The fuzzy logic (rule-based) expert component, depicted in Figure 5-5, combines the information received from the classification and model-based components. This component is used to assess the system condition. Having such a component reduces the impact of false alarms and enables diagnosis of unknown failure modes.

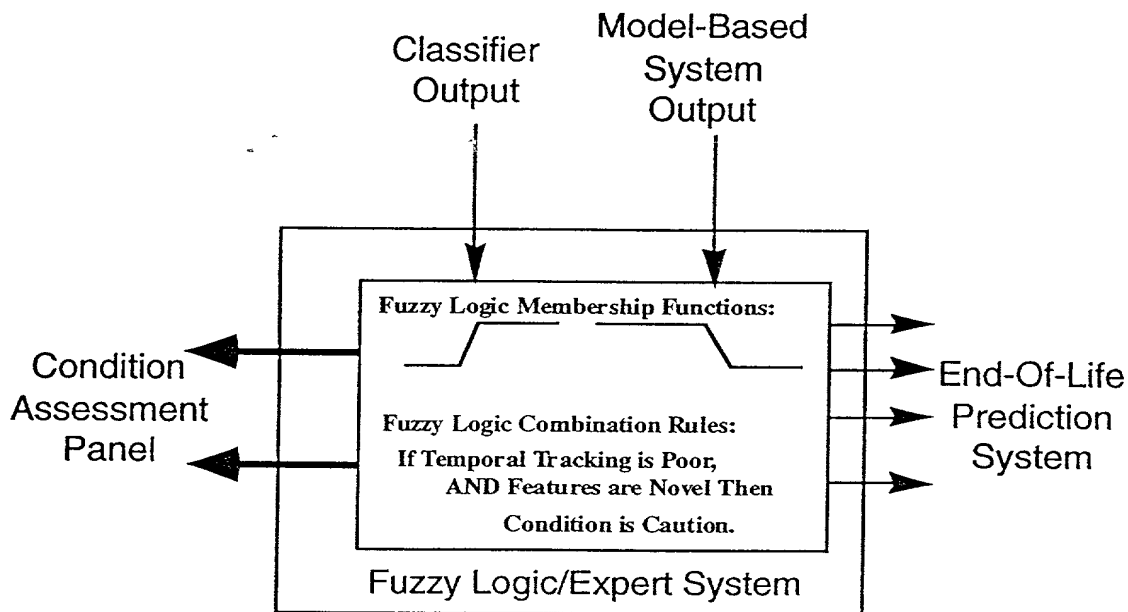


Figure 5-5 Fuzzy Logic/Expert Component Block Diagram.

Subsection 6. Condition Assessment Panel

The condition assessment panel, depicted in Figure 5-6, displays the condition information generated by the fuzzy logic rule-based expert component. This component is used to communicate: (1) the existence or lack of a fault, (2) the severity of fault, and, (3) the type of a fault.

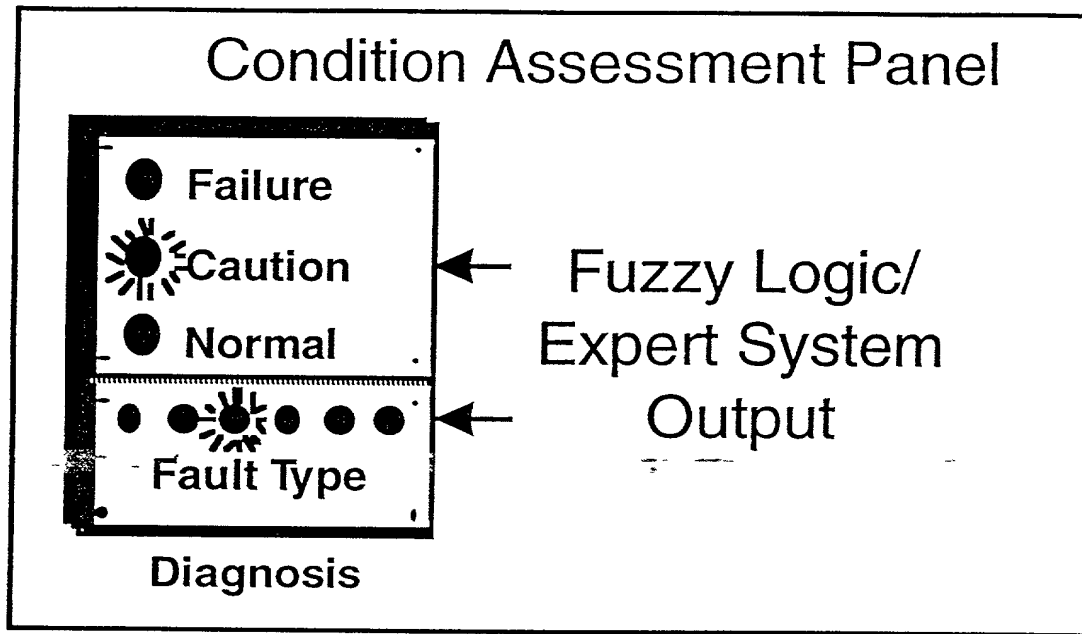


Figure 5-6 Condition Assessment Panel.

Section 3. END-OF-LIFE PREDICTION SUBSYSTEM

The end-of-life prediction subsystem is composed of three components which work as a whole to prognosticate the expected useful life of the equipment being assessed. The specific components that make-up the end-of-life prediction system are now described in more detail.

Subsection 1. Condition Prediction and End-Of-Life Prediction Component

The condition prediction end-of-life prediction component, depicted in Figure 5-7, combines the information received from the condition assessment subsystem and the data capture and preprocessing component. This component is used to prognosticate (predict) the system expected end-of-life using the currently assessed system condition and the impact of various indicators on the expected end-of-life. This component can be implemented using the signal processing inventions of TAMUS 1097. Implementations of this component using other signal processing algorithms are feasible.

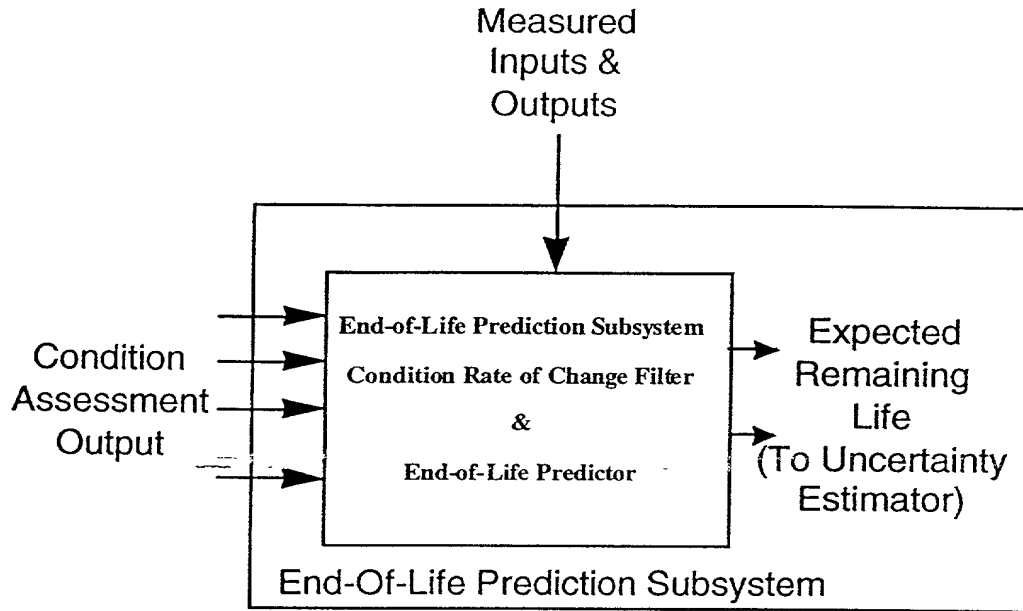


Figure 5-7 Condition Prediction and End-Of-Life Prediction Component Block Diagram.

Subsection 2. Predicted Condition and End-Of-Life Uncertainty Estimation Component

The predicted condition and end-of-life uncertainty estimation component, depicted in Figure 5-8, processes the information received from the condition prediction and end-of-life prediction component to obtain an estimate of the uncertainty in the expected predicted condition and end-of-life. As the actual end-of-life of the system approaches, the uncertainty computed by this subsystem decreases. This component can be implemented using the signal processing inventions of TAMUS 1097. Implementations of this component using other signal processing algorithms are feasible.

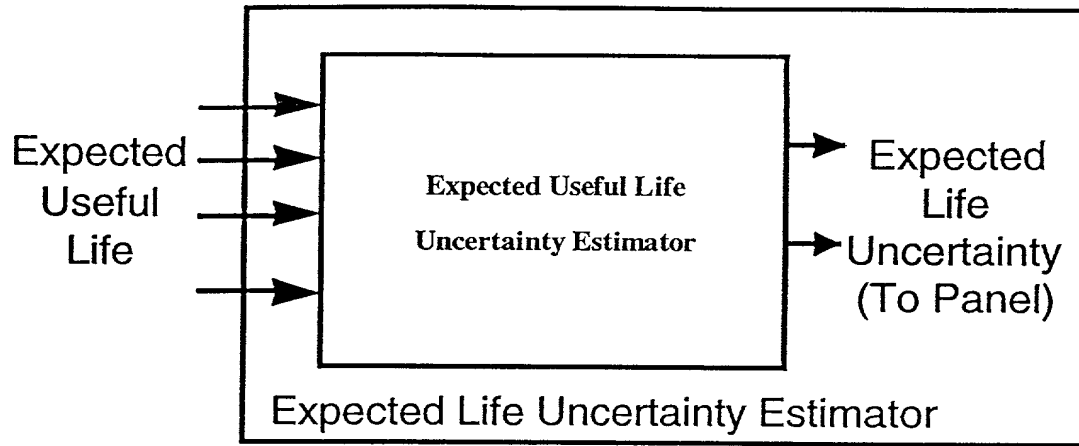


Figure 5-8 Predicted Condition and End-Of-Life Uncertainty Estimation Block Diagram.

Subsection 3. End-Of-Life Panel

The end-of-life prediction panel, depicted in Figure 5-9, displays the prognosis information generated by the end-of-life prediction and end-of-life uncertainty estimation subsystems. This subsystem is used to communicate: (1) the expected end-of-life, (2) the rate of deterioration of the expected end-of-life, and, (3) the uncertainty in the expected end-of-life.

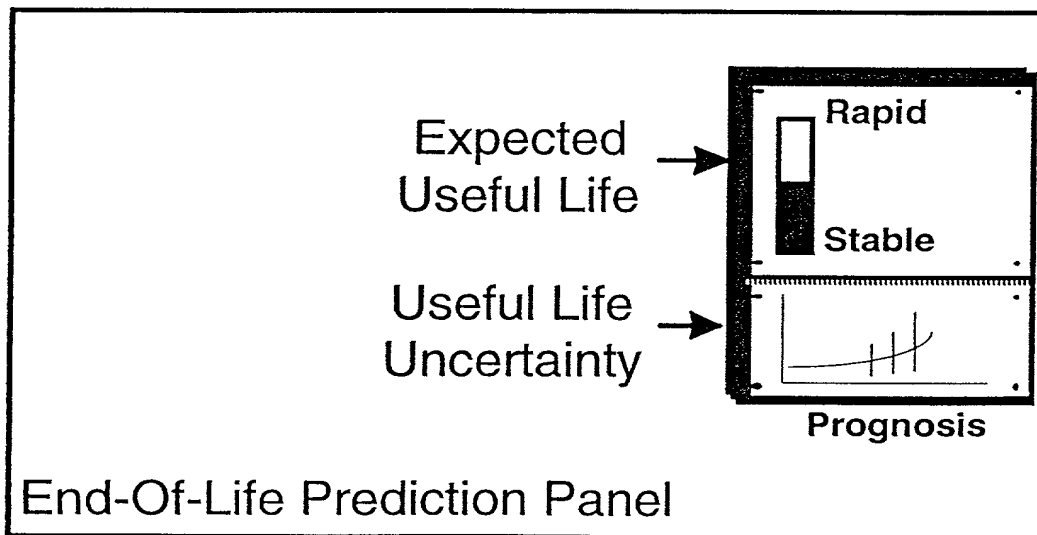


Figure 5-9 End-Of-Life Prediction Panel.

tion 4. CONDITION ASSESSMENT AND END-OF-LIFE PREDICTION OF MECHANICALLY COUPLED SYSTEMS

Once ICAPS is used to assess the condition and predict the end-of-life of a system, then it can be used to assess the condition and predict the end-of-life of another system to which it is coupled, using the first system as a transducer. This coupling could be mechanical or otherwise. This is depicted in figure 5-10.

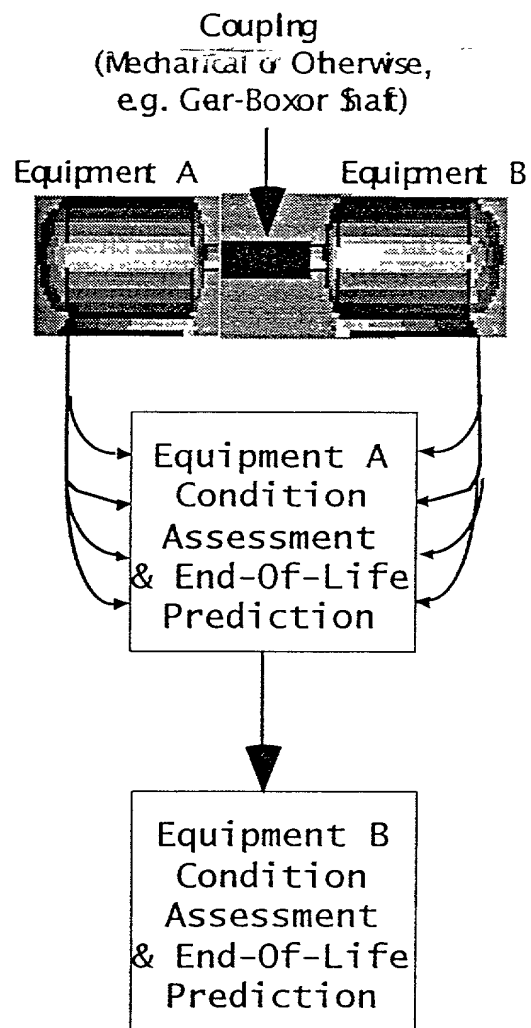


Figure 5-10 ICAPS for Coupled Systems.

Section 5. ICAPS IMPLEMENTATION - CONDITION ASSESSMENT SUBSYSTEM FOR MULTIPHASE AC INDUCTION MOTOR

Now we describe the application of the JIT Maintenance technology presented in the previous sections to a multiphase AC induction motor. The technology is tested on the virtual testbed, also described in this chapter, and on actual motor test data and some comparative results are presented.

The condition assessment system is composed of a number of subsystems which work as a whole to diagnose the current condition of the equipment being assessed. The specific subsystems that make-up the condition assessment system are now described in more detail.

Subsection 1. Data Capture and Preprocessing Component

The Data Capture and Preprocessing Component, depicted in Figure 5-11, is the interface between

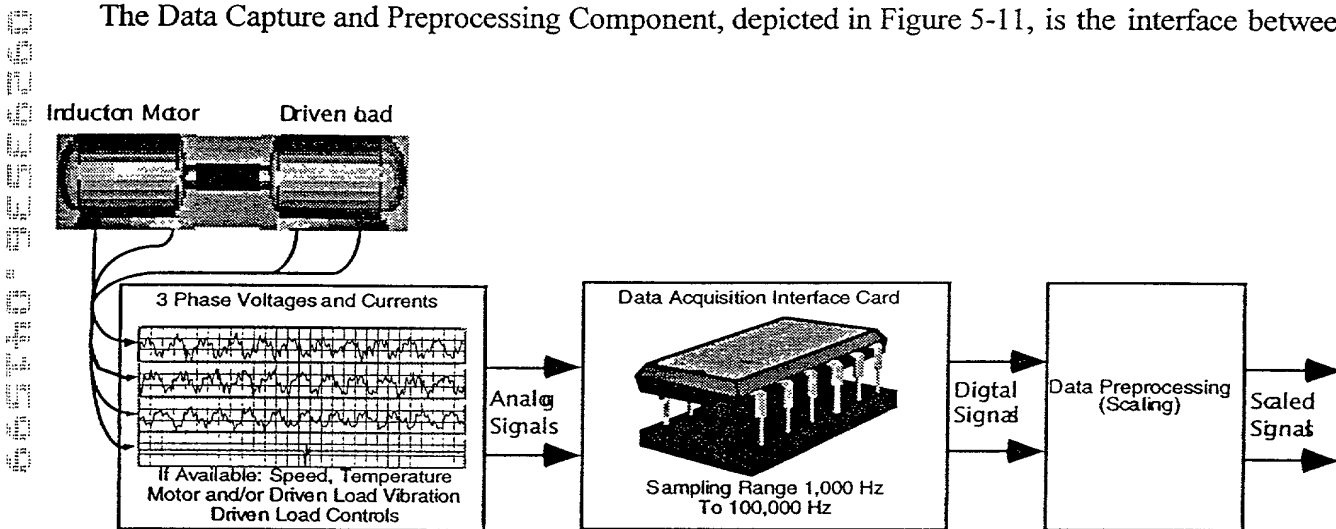


Figure 5-11 Motor Data Capture and Preprocessing Component Block Diagram.

the analog signals of the real world and the digital signals of the computer (“virtual”) world. This subsystem is implemented in hardware, for example using DAQ cards, and it converts the many analog signals, both motor inputs and outputs, it receives into digital signals via sampling. Prior to sampling the analog signals could be conditioned using analog filtering techniques to remove high frequency components or specific low frequency components. The sampling rate can be varying ranging up to 100 kHz or even 1 MHz.

Prior to sampling the signals are conditioned in analog form via the use of anti-aliasing low-pass filters, or other means of conditioning. Following sampling, the signals are conditioned and they are prepared for processing by the rest of the subsystems. The digital signal conditioning might take the form of additional digital filtering or scaling using some a priori known formulas. The sampled signals are the three-phase voltages and three-phase currents, but they could also contain the rotational speed, the rotational acceleration, the lateral acceleration, the motor temperature, etc..

Subsection 2. Model-Based Component

The model-based component, depicted in Figure 5-12, serves two broad purposes: (1) to decouple the effects of the motor inputs and disturbances, that is the three-phase voltages and load torque, on the motor outputs, the three-phase currents, and (2) to enable novelty detection for faults that have not been previously encountered.

The component consists of two subcomponents: (1) a filter to estimate the motor mechanical speed or torque when it cannot be directly measured, (2) a predictor for predicting the expected (or "healthy") motor response. Both the filter and the predictor could be based: (1) on first-principles only, with some parameter tuning, such as using the d-q-0 model, (2) on purely empirical (or adaptively obtained) models, such as using the neural networks of the JIT maintenance technology, or (3) hybrid combination of the above two options. The filter of the model-based component can be implemented using the signal processing inventions of TAMUS 1084, and the predictor of the model-based component can be implemented using the signal processing inventions of TAMUS 1058. Implementations using other signal processing algorithms are also feasible.

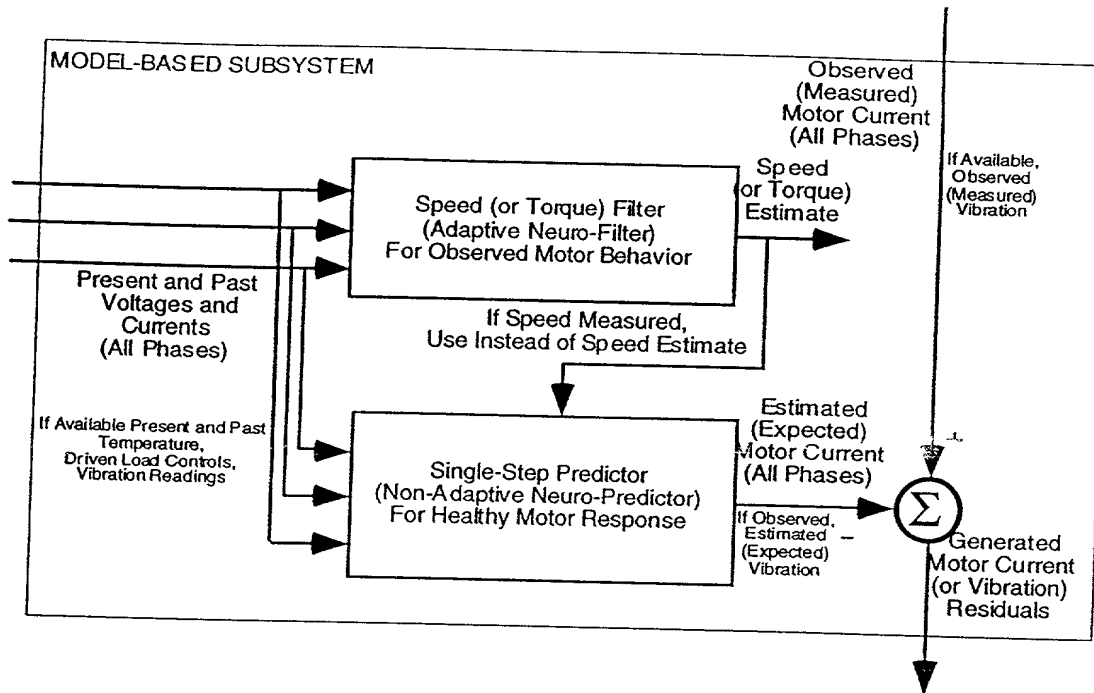


Figure 5-12 Model-Based Component Block Diagram.

Subsection 3. Signal-Based Component

The signal-based component, depicted in Figure 5-13, processes information from the model-based component outputs, the residuals. This component could also process information, in parallel, from the data capture and preprocessing component should that be necessary.

The signal-based component extracts spatio-temporal features related to faults. The expected signatures are canceled by the model-based subsystem, therefore the signatures (features) extracted by the signal-based subsystem belong to some predefined fault class.

The signal-based component can be implemented using any of the available algorithms in the literature. The use of the signal-based component in combination with the model-based component, as done in ICAPS, and the use of the signal-based component in combination of the rule-based system, as done in ICAPS, gives a unique implementation.

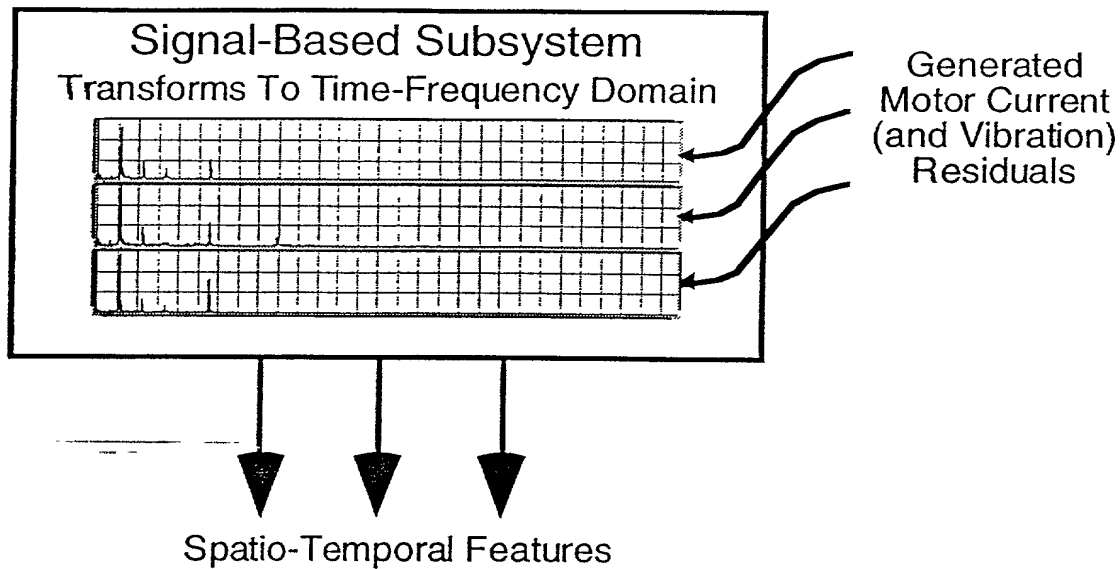


Figure 5-13 Signal-Based Component Block Diagram.

Subsection 4. Classification Component

The multi-stage classification component, depicted in Figure 5-14, processes information received from the signal-based component. This component is used to classify known failure modes. Having decoupled the features of “healthy” systems, the accuracy of the classification component is increased.

The classification component can be implemented using any of the available algorithms in the literature. The use of the model-based component outputs as inputs to the classification component, as done in ICAPS, gives a unique implementation.

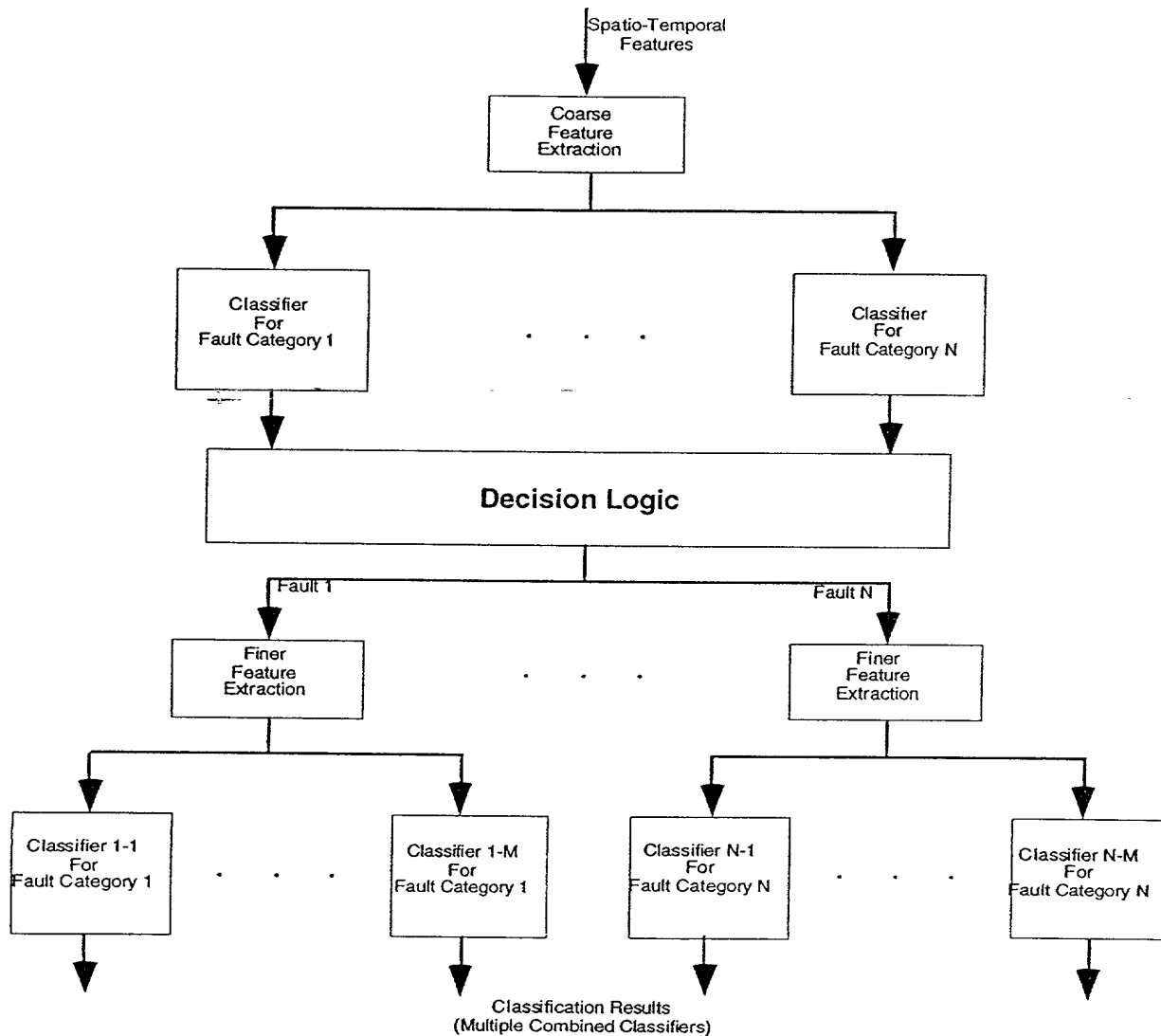


Figure 5-14 Multi-Stage Classification Component Block Diagram.

Subsection 5. Fuzzy Logic Expert Component

The fuzzy logic (rule-based) expert subsystem, depicted in Figure 5-15, combines the information received from the classification and model-based components. This component is used to assess the motor condition. Having such a subsystem reduces the impact of false alarms and enables diagnosis of unknown failure modes.

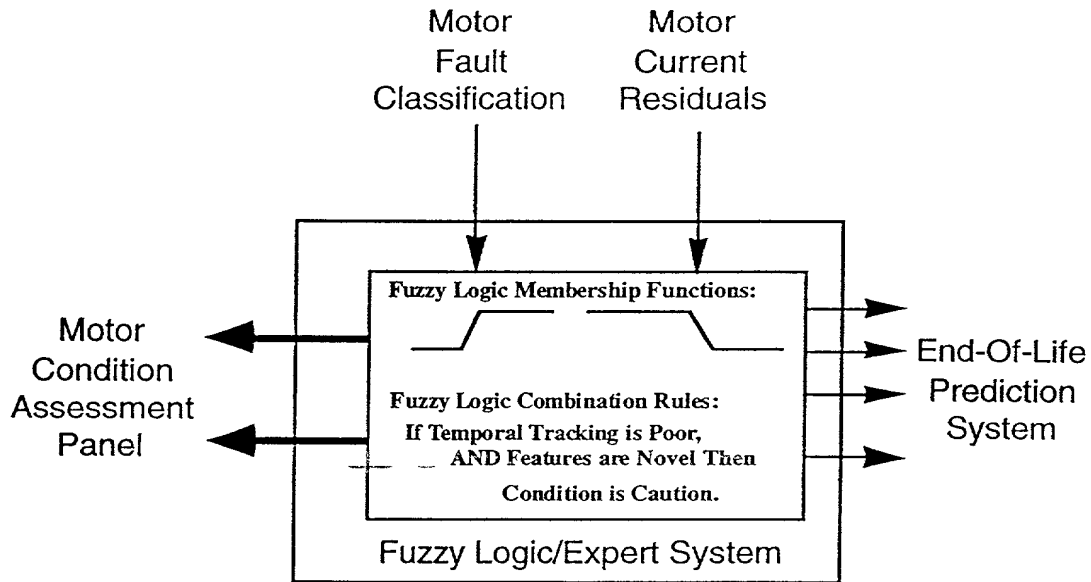


Figure 5-15 Fuzzy Logic/Expert Component Block Diagram.

Subsection 6. Condition Assessment Panel

The condition assessment panel, depicted in Figure 5-16, displays the motor condition information

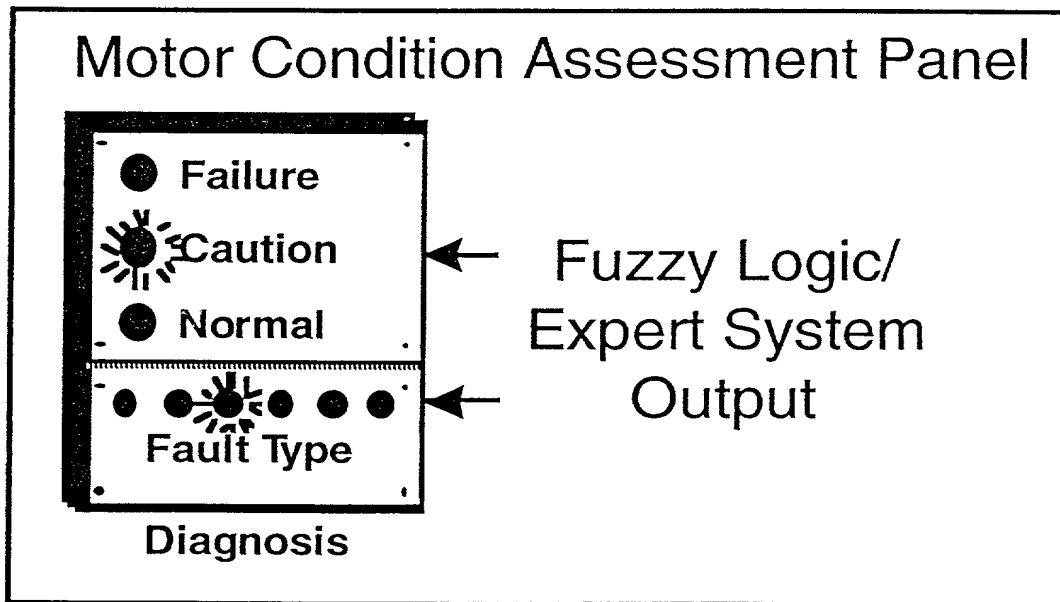


Figure 5-16 Motor Condition Assessment Panel.

generated by the fuzzy logic rule-based expert component. This component is used to communicate: (1) the existence or lack of a fault, (2) the severity of fault, and, (3) the type of a fault.

Section 6. ICAPS IMPLEMENTATION - END-OF-LIFE PREDICTION SUBSYSTEM FOR MULTIPHASE AC INDUCTION MOTOR

The end-of-life prediction system is composed of three subsystems which work as a whole to prognosticate the expected useful life of the motor being assessed. The specific subsystems that make-up the end-of-life prediction system are now described in more detail.

Subsection 1. Condition Prediction and End-Of-Life Prediction Component

The condition prediction end-of-life prediction component, depicted in Figure 5-17, combines the information received from the condition assessment subsystem and the data capture and preprocessing component. This component is used to prognosticate (predict) the motor expected end-of-life using the currently assessed motor condition and the impact of various indicators on the expected end-of-life, such as electric power quality, motor ambient temperature, and motor load torque pulsations. This component

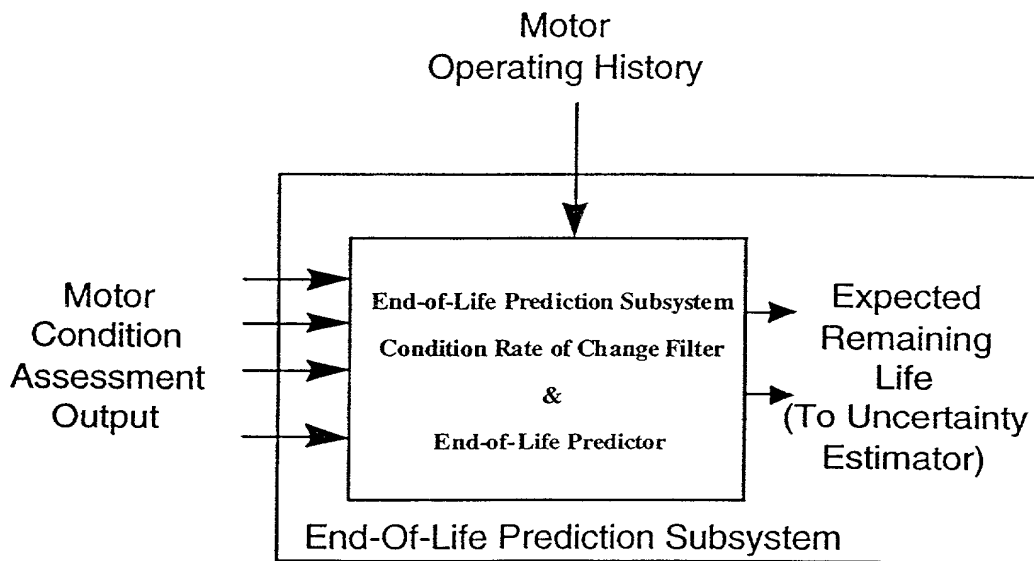


Figure 5-17 Condition Prediction and Motor End-Of-Life Prediction Component.

can be implemented using the signal processing inventions of TAMUS 1097. Implementations of this component using other signal processing algorithms are feasible.

Subsection 2. Predicted Condition and End-Of-Life Uncertainty Estimation Subsystem

The predicted condition and end-of-life uncertainty estimation component, depicted in Figure 5-18, processes the information received from the condition prediction and end-of-life prediction component to obtain an estimate of the uncertainty in the expected motor end-of-life. As the actual end-of-life of the motor approaches, the uncertainty computed by this subsystem decreases. This component can be

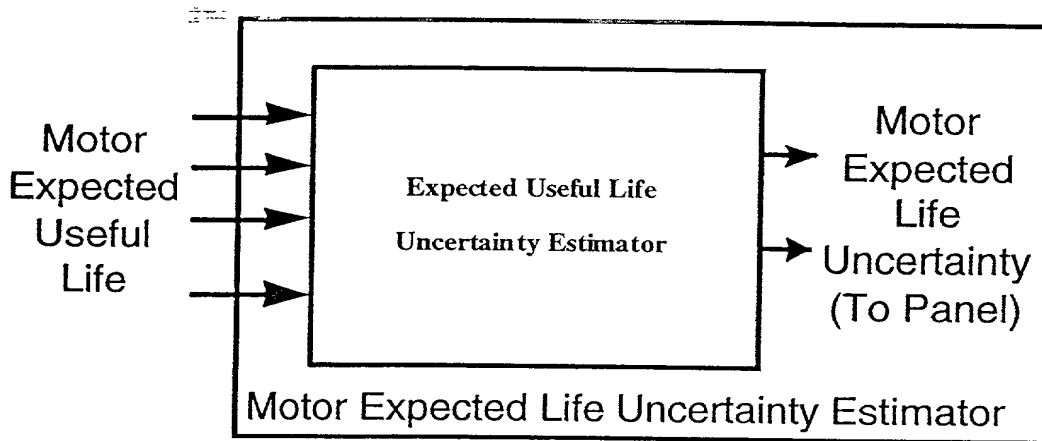


Figure 5-18 Predicted Condition and End-Of-Life Uncertainty Estimation Subsystem.

implemented using the signal processing inventions of TAMUS 1097. Implementations of this component using other signal processing algorithms are feasible.

Subsection 3. End-Of-Life Panel

The end-of-life prediction panel, depicted in Figure 5-19, displays the prognosis information generated by the end-of-life prediction and end-of-life uncertainty estimation subsystems. This subsystem is used to communicate: (1) the expected motor end-of-life, (2) the rate of deterioration of the expected motor end-of-life, and, (3) the uncertainty in the expected motor end-of-life.

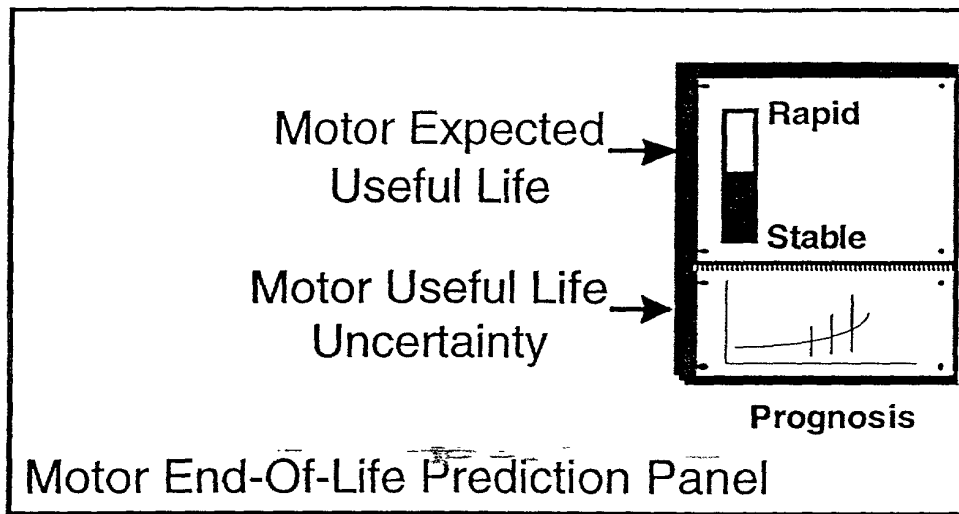


Figure 5-19 Motor End-Of-Life Panel.

Section 7. LOAD CONDITION ASSESSMENT AND END-OF-LIFE PREDICTION

Once ICAPS is used to assess the motor condition and predict the motor end-of-life, then it can be used to assess the condition and predict the end-of-life of the motor-driven equipment, such as a pump, compressor, fan, etc., using ICAPS implemented for such an equipment. The coupling could be mechanical or otherwise. This is depicted in Figure 5-20.

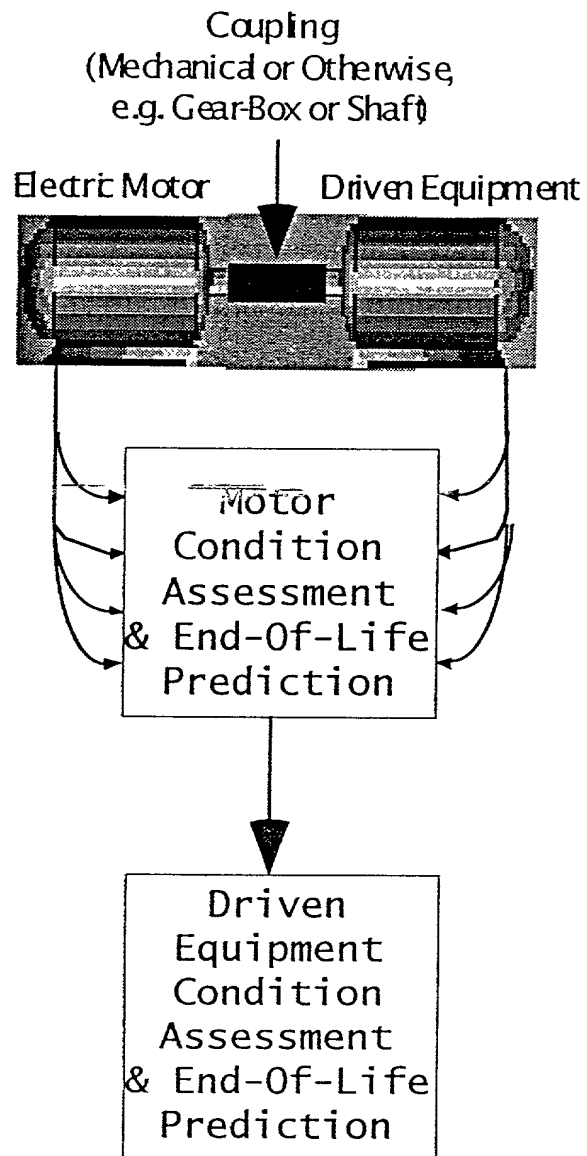


Figure 5-20 ICAPS for Motor-Load Combinations.

Section 8. VIRTUAL TESTBED FOR ELECTRIC MOTOR SYSTEMS

The aforementioned application of the JIT maintenance technology has been tested using the virtual motor system testbed shown in Figure 5-21. A similar virtual generator system testbed is shown in Figure 5-22.

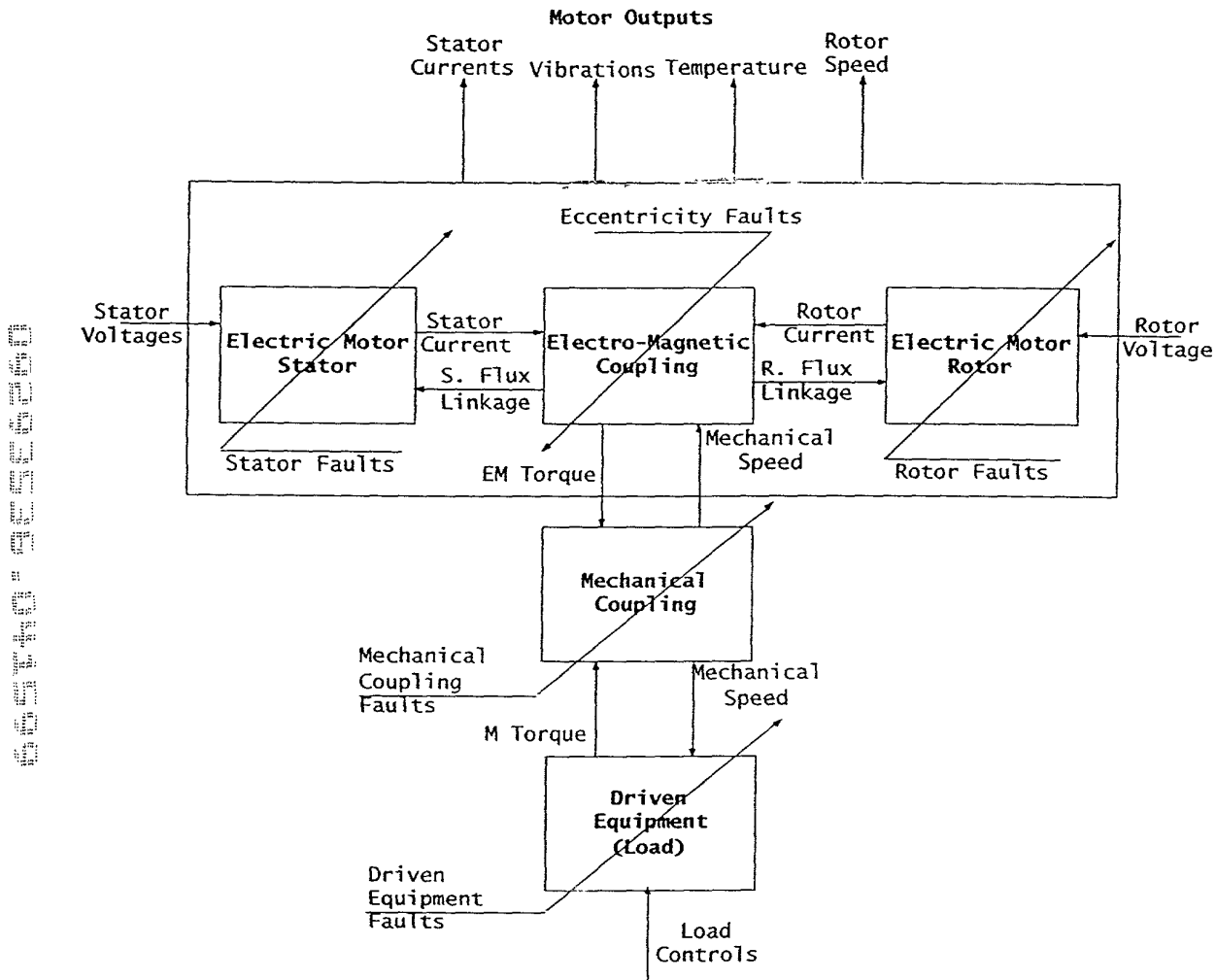


Figure 5-21 Virtual Testbed for Electric Motor Systems.

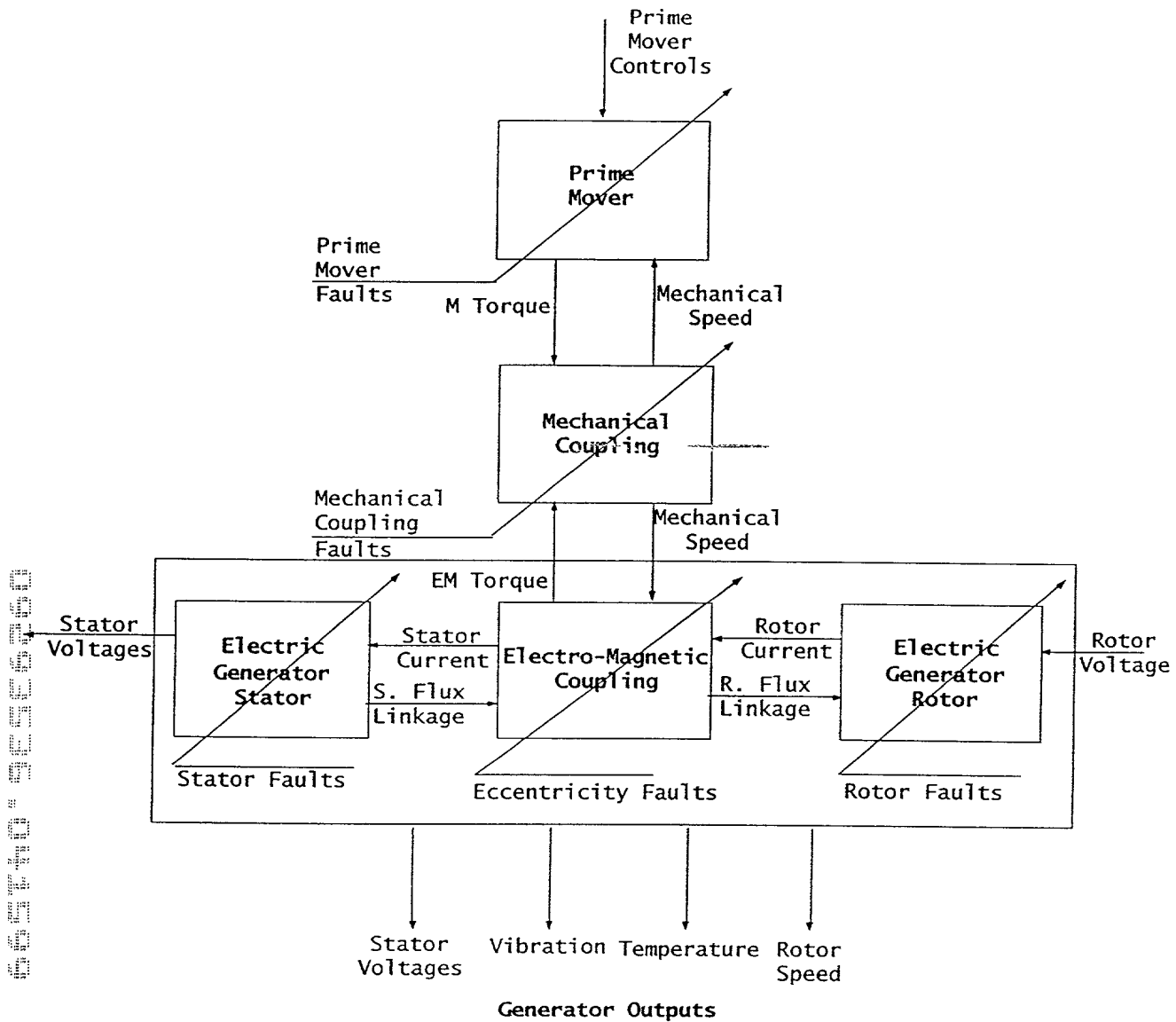


Figure 5-22 Virtual Testbed for Electric Generator Systems.

Section 9. ICAPS IMPLEMENTATION - TECHNOLOGY DEMONSTRATION AND COMPARISON RESULTS FOR MULTIPHASE AC INDUCTION MOTOR

The JIT maintenance technology application to an AC induction motor are now presented and compared with existing technology. First, we present the case of healthy motor driving a pump that creates pulsations. Using existing technology to diagnose this motor, one would obtain the motor current

spectrum shown in Figure 5-23. It can be seen that the current side-bands are present indicating broken bars, when in reality the motor is in healthy condition. The same scenario is now analyzed using the JIT maintenance technology. The results are shown on Figure 5-24. It can be seen that the current spectrum correctly shows the existence of no motor faults.

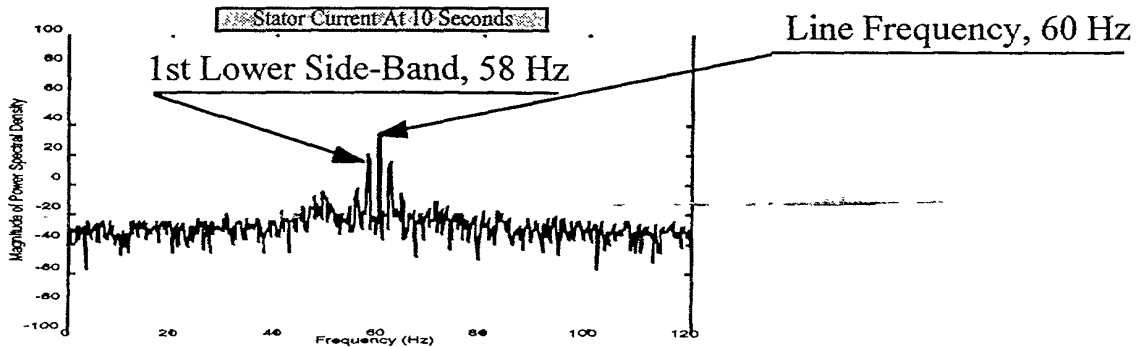


Figure 5-23 Broken Rotor Bars False Alarm With Existing Technology.

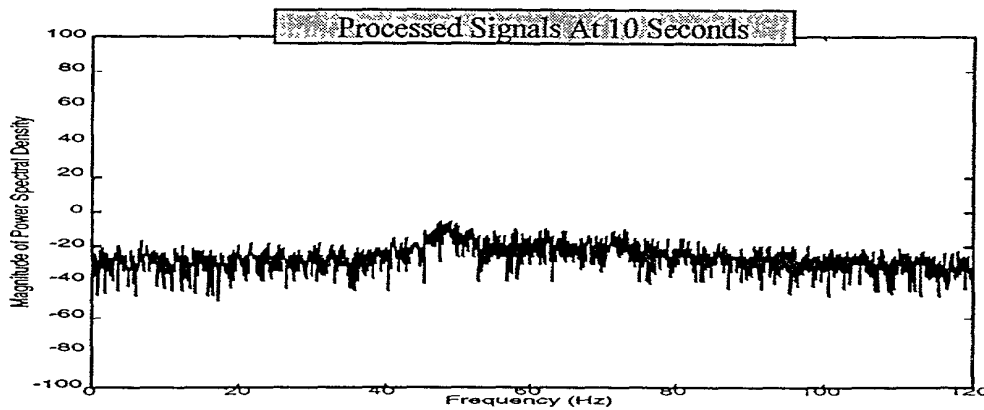


Figure 5-24 Correct Diagnosis with JIT Maintenance Technology.

Similar scenarios regarding rotor eccentricity and stator faults have been obtained demonstrating the effectiveness and superior performance of the JIT maintenance technology.

Chapter 6 VIRTUAL INSTRUMENTS FOR IMPLEMENTATION OF ICAPS

In this chapter the details of virtual instruments used in implementing the ICAPS are described. The virtual instruments described in here can be used for condition assessment and end-of-life prediction in conjunction with algorithms beyond those detailed in ICAPS. The invention contained in this chapter was first disclosed in TAMUS 1161.

Section 1. VIRTUAL CONDITION INSTRUMENT

A virtual equipment condition instrument is defined to be a software system that is connected to a physical piece of equipment through physical (or hardware) sensors and which can accurately, continuously, non-intrusively, and in real-time or in near real-time provide equipment condition information, i.e. provide equipment condition information without the need to disrupt equipment operation and without human intervention. If a sensor measures and displays the condition of equipment then it must be a virtual sensor. A condition sensor is by definition a virtual (or software implemented) sensor because condition (much like quality and unlike temperature or speed or pressure) is an attribute which cannot be directly measured. A condition sensor for a piece of equipment can be constructed by using a graphical programming language, like Visual Basic or the LabVIEW G-language. Nevertheless, a virtual sensor can also be constructed by programming it directly using a language such as C or C++. The front-end of one embodiment of the graphical user interface for a virtual equipment condition instrument is shown in Figure 6-1.

A virtual condition sensor for a piece of equipment is a software system that displays the following (non-exhaustive) list of directly measured or inferred quantities:

(A) Assessed Failure Information

- Failure mode class that might be occurring (including the lack of failure mode, i.e. healthy operation mode) and a detailed description of the failure mode, i.e. whether a specific sub-failure mode is occurring and whether there are multiple failures and how many, designated by color-coded displays,

- Severity of the failure mode, displaying the failure progression stage via color-coded information, i.e. whether Healthy, Warning or Failure,
- Urgency of the failure mode, displaying how fast the dominant failure mode(s) are developing via color-coded information, i.e. whether slow or fast using a quantifiable scale,
- If multiple failure modes are occurring simultaneously, the aforementioned failure-related information must be displayed for each assessed failure mode,
- The aforementioned failure information can be displayed in a number of ways, including but not limited to (i) number, (ii) bar (thermometer), (iii) color-coded, (iv) graphical, (v) text, and (vi) sound,
- The aforementioned failure information can be displayed on a variety of media, such as: (i) a CRT, (ii) other monitors, (iii) a gauge, (iv) an LCD, (v) an LED.

(B) Environmental and Operating Conditions - Fault Confusion Factors

- Information regarding the various environmental and operating conditions that might be instrumental in increasing the confidence level of the assessed condition. These are indicators that frequently confuse the condition assessment process,
- In the specific case of an electric motor, the quality of the power supply and the variations of the driven load (or mechanical speed) must be displayed. These two variables help improve the confidence in the assessed equipment condition.

(C) Failure Uncertainty Information

- For each of the failures assessed, an uncertainty level must be displayed in the form of a confidence interval,
- The confidence interval can be bound statistically, dynamically, or model-based.

(D) Efficiency and Other Performance Information

- The efficiency of the equipment in its present condition must be computed and displayed. Any one or a combination of the methods used in operational efficiency calculations is acceptable,

- For the computed efficiency, an uncertainty level must be displayed in the form of a confidence interval.

(E) Maintenance (or Repair) Information

- The recommended repairs for the assessed equipment condition must be displayed,
- A priority list must be displayed for the recommended repairs, and,
- A schedule estimate must be displayed for the recommended repairs.

(F) Cost (or Savings) Information

- The cost associated with deteriorating equipment efficiency must be displayed, on a per-unit and cumulative basis,
- The cost associated with the recommended repairs must be displayed, and,
- The cost associated with the down-time to perform the recommended repairs must be displayed.

(G) Historical Information

- All information (A) through (F) should be displayed in historic form, i.e. in the form of a time-series.

Furthermore, the virtual equipment condition instrument must provide sufficient real-time signal processing capabilities, such as filtering, windowing, etc., to enable a user to trade-off the various factors influencing the (accuracy) of the displayed equipment condition information in real-time.

Figure 6-1 Virtual Equipment Condition Instrument.

Section 2. VIRTUAL END-OF-LIFE INSTRUMENT

A virtual equipment end-of-life instrument is defined to be a software system that is connected to a physical piece of equipment through physical (or hardware) sensors and which can accurately, continuously, non-intrusively, and in real-time or in near real-time provide equipment end-of-life information, i.e. provide equipment end-of-life information without the need to disrupt equipment operation and without human intervention. If a sensor measures and displays the end-of-life of equipment then it must be a virtual sensor. An end-of-life sensor is by definition a virtual (or software implemented) sensor because end-of-life (much like condition, quality and unlike temperature or speed or pressure) is an attribute which cannot be directly measured. An end-of-life sensor for a piece of equipment can be constructed by using a graphical programming language, like Visual Basic or the

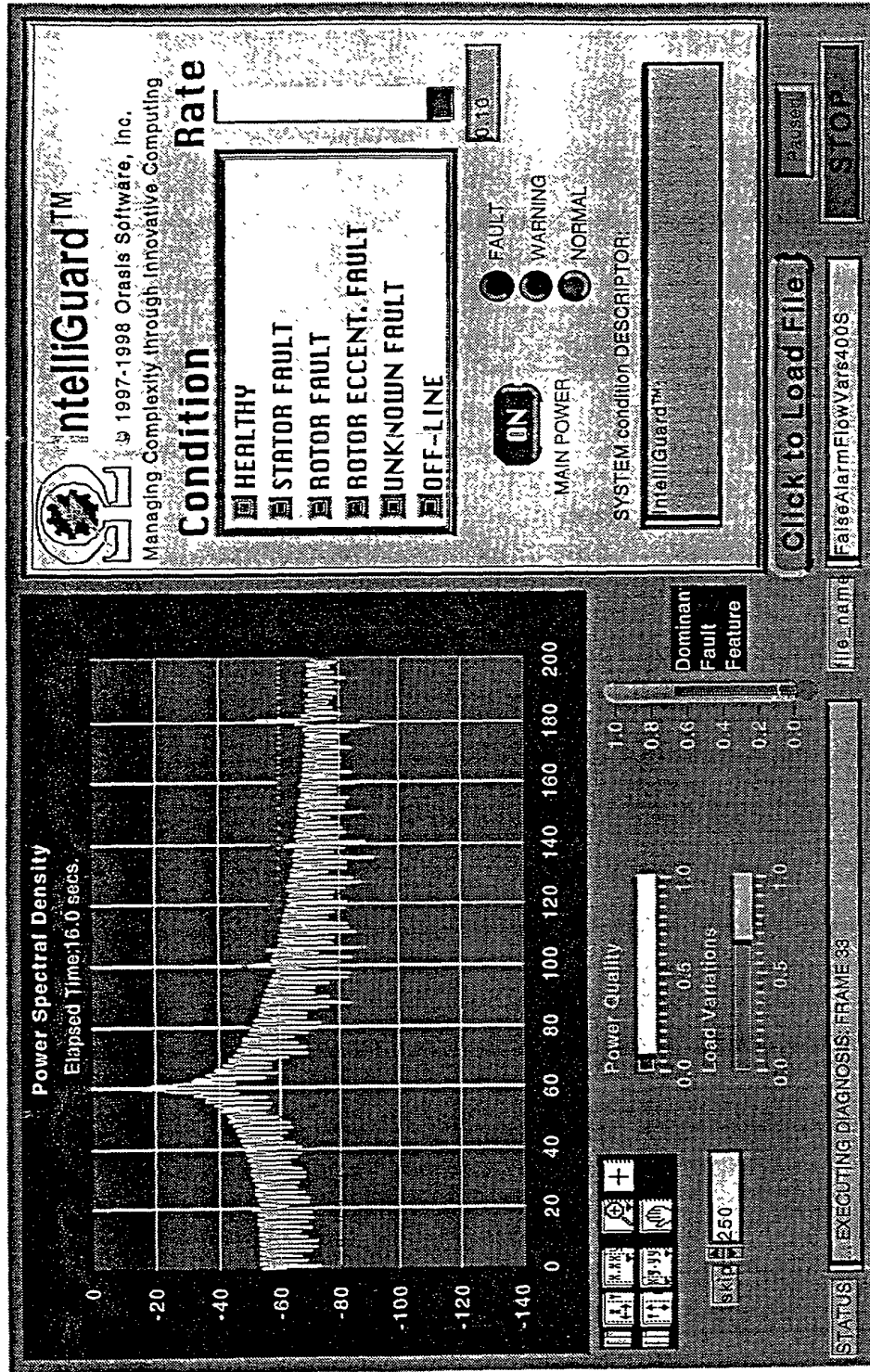


FIGURE 6-1

LabVIEW G-language. Nevertheless, a virtual sensor can also be constructed by programming it directly using a language such as C or C++.

A virtual end-of-life sensor for a piece of equipment is a software system that displays the following (non-exhaustive) list of directly measured or inferred quantities:

(A) Predicted Failure Information - Condition Forecast

- Failure mode class that is predicted to occur (including the lack of failure mode, i.e. healthy operation mode) and a detailed description of the anticipated failure mode, i.e. whether a specific sub-failure mode will be occurring and whether there will be multiple failures and how many, designated by color-coded displays,
- Severity of the predicted failure mode, displaying the failure progression stage via color-coded information, i.e. whether Healthy, Warning or Failure,
- Urgency of the predicted failure mode, displaying how fast the dominant failure mode(s) are developing via color-coded information, i.e. whether slow or fast using a quantifiable scale,
- If multiple failure modes are predicted simultaneously, the aforementioned failure-related information must be displayed for each assessed failure mode,
- The aforementioned failure information can be displayed in a number of ways, including but not limited to (i) number, (ii) bar (thermometer), (iii) color-coded, (iv) graphical, (v) text, and (vi) sound,
- The aforementioned failure information can be displayed on a variety of media, such as: (i) a CRT, (ii) other monitors, (iii) a gauge, (iv) an LCD, (v) an LED.

(B) End-of-Life Information

- The expected useful (or operational) remaining life of the equipment (in hours, days, weeks, moths, etc.); this can also be interpreted as the mean-time before failure and it can be computer in a variety of methods, such as statistically based on historical data-bases, based on current condition, etc.,

- The uncertainty in the expected useful (or operational) remaining life of the equipment (in hours, days, weeks, months, etc.),
- The confidence level in the expected useful (or operational) remaining life of the equipment (in either standard deviations or %),

(C) Predicted Failure Uncertainty Information

- For each of the failures predicted, an uncertainty level must be displayed in the form of a confidence interval,
- The confidence interval can be bound statistically, dynamically, or model-based.

(D) Efficiency and Other Performance Information

- The predicted efficiency of the equipment must be computed and displayed. Any one or a combination of the methods used in operational efficiency calculations is acceptable,
- For the predicted efficiency, an uncertainty level must be displayed in the form of a confidence interval.

(E) Maintenance (or Repair) Information

- The recommended repairs for the predicted equipment condition must be displayed,
- A priority list must be displayed for the predicted repairs to be recommended, and,
- A schedule estimate must be displayed for the predicted repairs to be recommended.

(F) Cost (or Savings) Information

- The cost associated with the predicted equipment efficiency must be displayed, on a per-unit and cumulative basis,
- The cost associated with the predicted repairs to be recommended must be displayed, and,
- The cost associated with the down-time to perform the predicted repairs to be recommended must be displayed.

(G) Historical Information

- All information (A) through (F) should be displayed in historic form, i.e. in the form of a time-series.

Furthermore, the virtual equipment end-of-life instrument must provide sufficient real-time signal processing capabilities, such as filtering, windowing, etc., to enable a user to trade-off the various factors influencing the (accuracy) of the displayed equipment condition information in real-time.

50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

Chapter 7 JIT MAINTENANCE TECHNOLOGY APPLICATIONS

In this chapter the applications of the ICAPS technology for JIT maintenance are listed. We start from rotating equipment and then generalize to non-rotating equipment.

Section 1. ROTATING EQUIPMENT

Table 7-1 presents a non-exhaustive list of rotating equipment for which the JIT maintenance technology is applicable. Furthermore, Table 7-2 presents a non-exhaustive list of specific electric motor types appropriate for the JIT maintenance technology. A generic block diagram describing the electric motor applications of the JIT maintenance technology is given in Figure 7-1. A similar block diagram for electric generator applications of JIT maintenance technology is shown in Figure 7-2.

Table 7-1 Rotating Equipment Applications.

Rotating Equipment Category	Specific Examples
Electric Machines	Motors and Generators
Turbomachinery	Turbines
Driven Equipment	Pumps, Compressors, Fans, etc.
Other Prime Movers	Diesel Engines, IC Engines, etc.

Table 7-2 Electric Motor Applications.

Electric Motors	
Motor Type	Motor Variations
Alternating Current (AC) Induction Motors	Single-Phase and Polyphase Squirrel Cage and Wound Rotor
AC Synchronous Motors	
Direct Current (DC) Motors	
Universal AC/DC Motors	
Permanent Magnet Motors	
Switched Reluctance Motors	

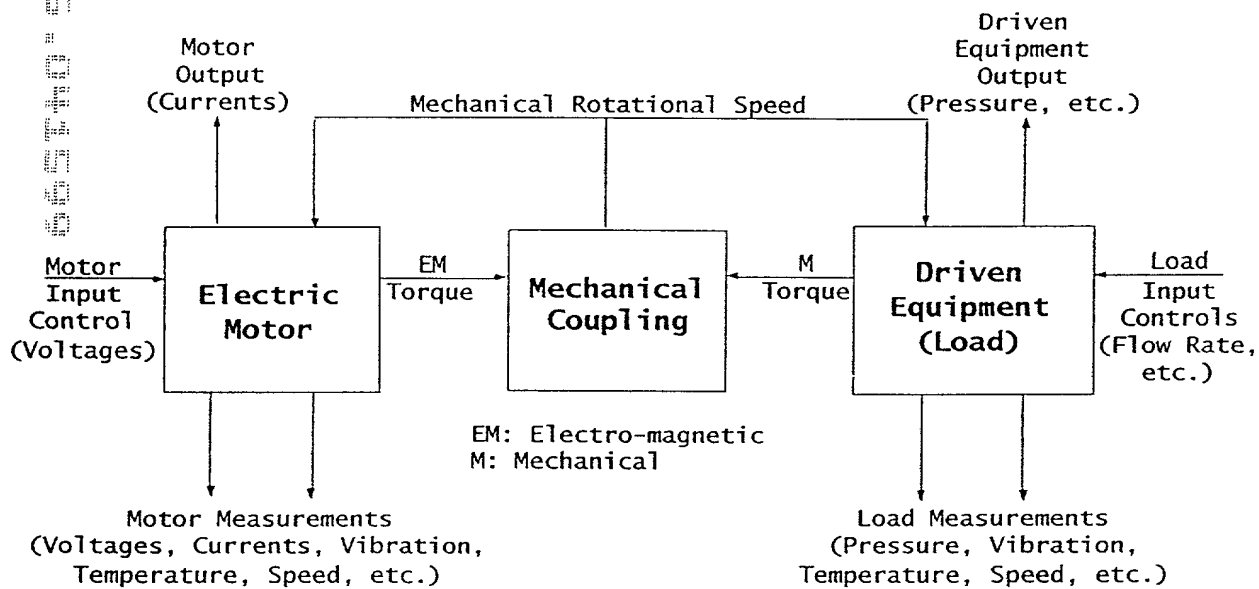


Figure 7-1 Electric Motor Applications.

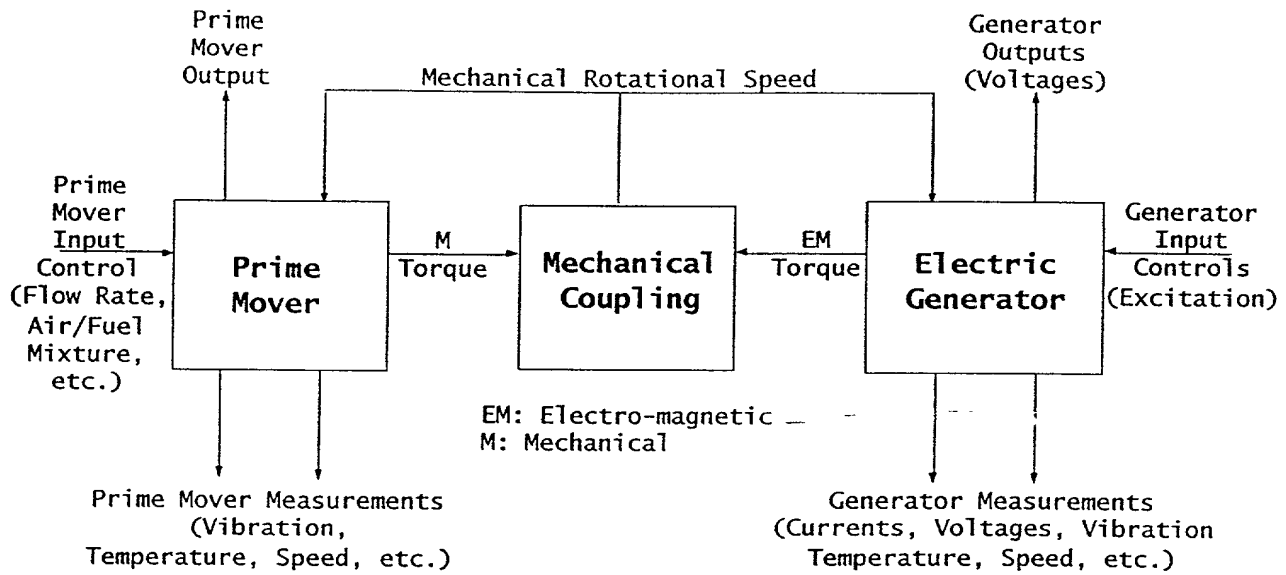


Figure 7-2 Candidate Generator Applications.

Section 2. OTHER EQUIPMENT

We now turn our attention to other (non-rotating) equipment, as depicted in Figure 7-3. Among the major applications are boilers, heat exchangers, and other process systems, such as chemical process systems. Additionally, electrical equipment, such as transformers and batteries, and mechanical equipment, such as valves and gear-boxes, are prime candidates.

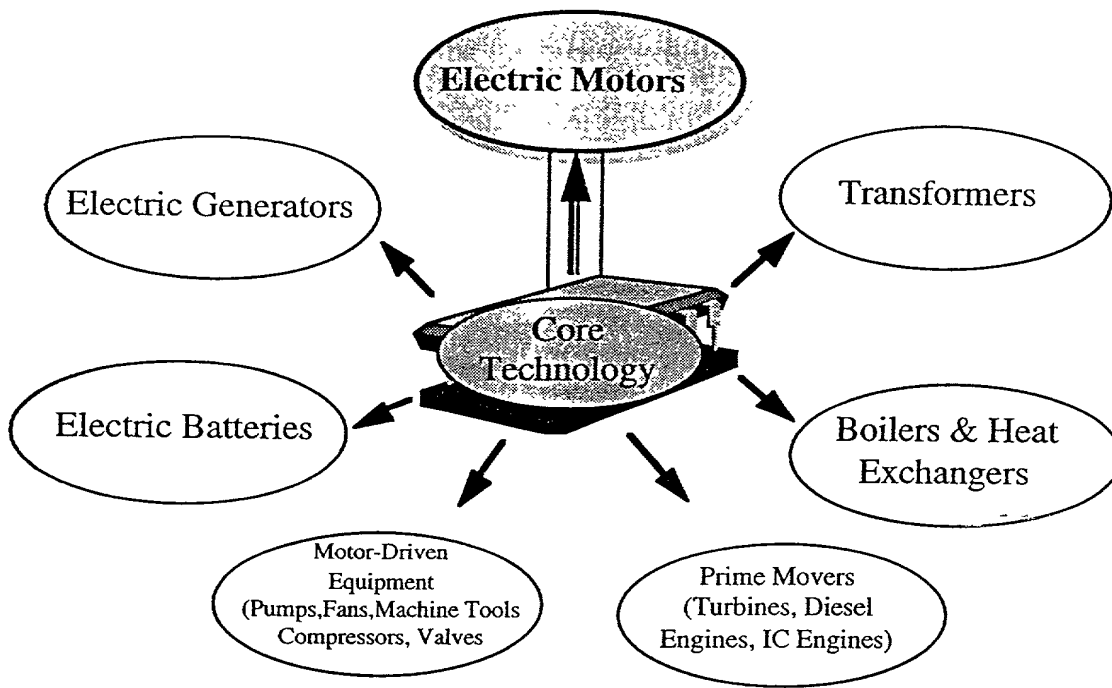


Figure 7-3 Application of Core Software Technology to Industrial Equipment.

ABSTRACT

Modeling Complex Process Systems Using The
Recurrent Multilayer Perceptron. (August 1995)

Omar Rais, B.S., Texas A&M University; M.S., Texas A&M University

Chair of Advisory Committee: Dr. Alexander G. Parlos

The objective of this research study is to develop a method for designing Computational Neural Networks (CNNs)-based empirical models capable of performing accurate multi-step-ahead (SSP) and single-step-ahead prediction (MSP) for complex process systems. Empirical models which can perform accurate MSP have many applications in the areas of forecasting and prediction, control engineering, and condition monitoring and fault diagnosis.

Conventional modeling methods utilizing first principles have many disadvantages, such as big development cost, potential inaccuracies in view of drifts, inability to incorporate tear and wear effects, to just name a few. Moreover, for complex process systems the conventional techniques of System Identification (SI) fail to give models capable of performing accurate MSP.

The proposed empirical modeling method makes use of the Recurrent Multilayer Perceptron (RMLP) neural network. The RMLP is a hybrid, feedforward and feedback, neural network which exhibits local information feedback through time-delayed recurrency and cross-talk. Two learning algorithms are considered when using the RMLP, the Teacher Forcing (TF) algorithm, and the Global Feedback (GF) algorithm. Both are based on a dynamic gradient descent approach. The difference

between the TF and the GF algorithms is that the former utilizes past sensed systems outputs, whereas the latter utilizes past RMLP predictions as inputs to the network. The proposed method also includes guidelines for choosing the appropriate network architecture and the training stopping criteria.

An artificial case-study, as well as a U-Tube Steam Generator (UTSG), were used to validate the accuracy of the proposed modeling method. For the artificial problem, the model based on conventional identification methods gave good results, while the CNN-based model gave slightly improved results. However, both models were capable of performing accurate MSP. For the UTSG, the CNN models performed well in MSP as well as in SSP. Validation studies using the developed CNN models have demonstrated that they exhibit substantial generalization of the operational UTSG dynamics. Models based on conventional identification methods failed to achieve such results. The success of the proposed modeling method is believed to be the results of the GF used in the RMLP which provides it with global and local memory, the associated dynamic learning algorithm, and the guidelines used in choosing the network architecture. The prediction results obtained when using TF in the RMLP were much better than the results obtained using conventional SI algorithms. However, these results were still inferior compared to the results obtained using GF in the RMLP. This research study and the accompanying results demonstrate that the proposed CNN design method and the associated algorithms provide a feasible and robust procedure for modeling complex process systems.

ACKNOWLEDGMENTS

The author would like to express his deepest gratitude to his thesis advisor, Dr. Alexander G. Parlos, for his endless support, guidance, concern and encouragement throughout the course of this research. The author would like also to thank Dr. Amir Atiya for providing the derivations of several neural network algorithms used in this research.

The author would like to thank all his colleagues. Sunil Menon, and Esmaeil Oufi for their support and for helping him write major parts of the codes used in this research. The author would like also to express heartfelt thanks to his parents Mr. Hadj Tahar Rais, and Mrs. Assia Harouchi for their moral support, and endless love.

TABLE OF CONTENTS

CHAPTER	Page
I INTRODUCTION	1
I.1 Brief Notes on Complex Systems	3
I.2 Motivation	4
I.3 Biological Neuron vs. Computational Neuron	7
I.4 Literature Review	11
I.5 Research Contributions	17
I.6 Organization of the Dissertation	18
II SYSTEM IDENTIFICATION AND ITS APPLICATIONS	20
II.1 Introduction	20
II.2 System Identification Procedures	22
II.3 Applications in Forecasting and Prediction	25
II.4 Applications in Control Engineering	26
II.5 Applications in Condition Monitoring and Fault Diagnosis	29
II.6 Limitations of Conventional Identification Algorithms	33
III THE RECURRENT MULTILAYER PERCEPTRON NETWORK AND ITS USE AS A NONLINEAR PREDICTOR	36
III.1 Introduction	36
III.2 Computational Neural Networks Architectures	38
III.3 Single-Step-Ahead and Multi-Step-Ahead Prediction	42
III.4 Recurrent Multilayer Perceptron Learning Algorithm with Teacher Forcing	51
III.5 Recurrent Multilayer Perceptron Learning Algorithm with Global Feedback	56
III.6 Network Architecture Design	64
III.7 Chapter Summary	69
IV A COMPARISON OF CONVENTIONAL AND COMPUTATIONAL NEURAL NETWORK APPROACHES IN NONLINEAR SYSTEM IDENTIFICATION	70
IV.1 Introduction	70
IV.2 The Polynomial Nonlinear AutoRegressive with eXogeneous Input Model Structure	71
IV.3 A Case-Study	78

CHAPTER	Page
IV.4 Chapter Summary	117
V U-TUBE STEAM GENERATOR EMPIRICAL MODELING	121
V.1 Introduction	121
V.2 Description of the U-Tube Steam Generator	122
V.3 Description of the Available UTSG Simulator	122
V.4 U-Tube Steam Generator Modeling	126
V.5 Computer Simulation Results and Model Validation	130
V.6 Chapter Summary	176
VI SUMMARY AND CONCLUSIONS	180
VI.1 Summary	180
VI.2 Conclusions and Recommendations	190
REFERENCES	192
APPENDIX	
A THE FUNCTIONAL NEURON	198
VITA	200

LIST OF TABLES

Table		Page
1	Relative Mean-Squared Errors for the Case-Study using the Polynomial NARX.	90
2	Relative Mean-Squared Errors for the Case-Study.	102
3	Relative Mean-Squared Errors for the Case-Study	114
4	UTSG Empirical Model Validation at Low Power Levels.	141
5	UTSG Empirical Model Validation at Medium Power Levels.	152
6	UTSG Empirical Model Validation at High Power Levels.	165

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

LIST OF FIGURES

Figure		Page
1	A Theoretical Action Potential of a Biological Neuron (Left), and an Experimental Action Potential of a Mammalian fiber (Right).	10
2	Stages of the System Identification Process.	23
3	Generic Architecture for Condition Monitoring and Fault Diagnosis . . .	32
4	The Feedforward Multilayer Perceptron Network.	39
5	The Recurrent Multilayer Perceptron Network.	40
6	Multi-Step-Ahead Prediction Schematic with $(p+1)$ -Step Horizon. . . .	43
7	Summary of Predictor Architectures.	47
8	An RMLP Utilized as a Teacher Forcing Predictor.	53
9	An RMLP with Global Feedback Utilized as a $(p+1)$ -Step-Ahead Predictor.	57
10	MSP within The Training Set.	58
11	First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	82
12	Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	83
13	First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	84
14	Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	85

Figure		Page
15	First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	86
16	Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	87
17	First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	88
18	Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	89
19	Mean Square Error of the Testing Set Using Global Feedback for Different Number of Delayed Outputs utilized in the Input Layer; Individual weight Update Mode.	93
20	First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	94
21	Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	95
22	First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	96
23	Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	97
24	First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	98

Figure	Page
25 Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	99
26 First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	100
27 Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	101
28 Mean Square Error of the Testing Set Using Global Feedback for Different Number of Delayed Outputs Utilized in the Input Layer.	104
29 First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	106
30 Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	107
31 First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	108
32 Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	109
33 First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	110
34 Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	111
35 First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	112

	Page
36 Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input ; Bottom: Pulse Input.)	113
37 A CNN which the Input Layer Consisting of Terms Obtained from the Polynomial NARX Model Structure Detection Algorithm.	118
38 Mean Square Error of the Best Networks, Utilizing CNN With Teacher Forcing.	119
39 Schematic Diagram of a U-Tube Steam Generator.	123
40 Block Diagram of the UTSG with Input, Disturbances, and Measured Outputs.	125
41 Polynomial NARX Water Level Response to a Step Power Level Increase from 20% to 25% of Full Power.	132
42 Polynomial NARX Water Level Response to a Ramp Power Level Decrease from 35% to 20% of Full Power.	133
43 Teacher Forcing CNN Water Level Response to a Step Power Level Increase from 20% to 25% of Full Power.	135
44 Teacher Forcing CNN Water Level Response to a Ramp Power Level Decrease from 35% to 20% of Full Power.	136
45 Computational Neural Network Predictor Used for Multi-Step-Ahead Prediction.	139
46 UTSG Transient Response Prediction for a 5% to 7.5% of Full Power Step.	142
47 UTSG Transient Response Prediction for a 7.5% to 10% of Full Power Step.	143
48 UTSG Transient Response Prediction for a 10% to 15% of Full Power Step.	144
49 UTSG Transient Response Prediction for a 15% to 20% of Full Power Step.	145

Figure	Page
50 UTSG Transient Response Prediction for a 5% to 20% of Full Power Ramp.	146
51 UTSG Transient Response Prediction for a 20% to 5% of Full Power Ramp.	147
52 UTSG Transient Response Prediction for a 5% to 15% of Full Power Ramp.	148
53 UTSG Transient Response Prediction for a 15% to 5% of Full Power Ramp.	149
54 UTSG Transient Response Prediction for a 20% to 10% of Full Power Ramp.	150
55 UTSG Transient Response Prediction for a 10% to 20% of Full Power Ramp.	151
56 UTSG Transient Response Prediction for a 20% to 25% of Full Power Step.	154
57 UTSG Transient Response Prediction for a 30% to 35% of Full Power Step.	155
58 UTSG Transient Response Prediction for a 40% to 45% of Full Power Step.	156
59 UTSG Transient Response Prediction for a 20% to 50% of Full Power Ramp.	157
60 UTSG Transient Response Prediction for a 50% to 20% of Full Power Ramp.	158
61 UTSG Transient Response Prediction for a 35% to 45% of Full Power Ramp.	159
62 UTSG Transient Response Prediction for a 45% to 35% of Full Power Ramp.	160
63 UTSG Transient Response Prediction for a 20% to 35% of Full Power Ramp.	161

Figure	Page
64 UTSG Transient Response Prediction for a 35% to 20% of Full Power Ramp.	162
65 UTSG Transient Response Prediction for a 20% to 30% of Full Power Ramp.	163
66 UTSG Transient Response Prediction for a 30% to 20% of Full Power Ramp.	164
67 UTSG Transient Response Prediction for a 50% to 55% of Full Power Step.	166
68 UTSG Transient Response Prediction for a 60% to 65% of Full Power Step.	167
69 UTSG Transient Response Prediction for a 80% to 85% of Full Power Ramp.	168
70 UTSG Transient Response Prediction for a 90% to 95% of Full Power Ramp.	169
71 UTSG Transient Response Prediction for a 50% to 95% of Full Power Ramp.	170
72 UTSG Transient Response Prediction for a 95% to 50% of Full Power Ramp.	171
73 UTSG Transient Response Prediction for a 60% to 80% of Full Power Ramp.	172
74 UTSG Transient Response Prediction for a 80% to 60% of Full Power Ramp.	173
75 UTSG Transient Response Prediction for a 50% to 75% of Full Power Ramp.	174
76 UTSG Transient Response Prediction for a 75% to 50% of Full Power Ramp.	175
77 UTSG Water Level Closed Loop Response Prediction to a Lost of Load from 100% to 5% of Full Power.	177

Figure	Page
78 UTSG Water Level Open Loop Response Prediction to a Steam Flow rate drop from 100% to 5% of Full Power.	178
79 UTSG Water Level Open Loop Response Prediction to a Feed Water Flow rate drop from 100% to 5% of Full Power.	179
80 A Functional Neuron (Perceptron)	199

UTSG Water Level Open Loop Response Prediction to a Steam Flow
rate drop from 100% to 5% of Full Power.

CHAPTER I

INTRODUCTION

The objective of this research study is to develop a method for designing Computational Neural Network (CNN) models capable of performing accurate multi-step-ahead (MSP) and single-step-ahead prediction (SSP) . The developed method is to be tested on a real-world problem, such as the U-Tube Steam Generator (UTSG). This field of study is known as system identification (SI) or empirical modeling. SI is the art of building models using input-output data sets collected from the system under study, while physical modeling is the art of building models using the laws of physics, such as conservation of mass, energy, and momentum equations. Identified system models, in general, have many applications in the areas of forecasting, prediction, control engineering, signal processing, and fault diagnosis, among others.

SI deals exclusively with the problem of building empirical models of systems based on observed data. Many theories have been developed in this area and they are known to perform well in some practical applications. The importance of SI has increased significantly in many fields, especially in engineering applications, where model building has become one of the most important and time consuming aspects of system design and operation. Design of accurate models is necessary for understanding the behavior of a system, and for their mathematical analysis. Without an adequate model of

This report follows a style based on the *IEEE Transactions on Automatic Control*.

the system to be controlled, for example, a good controller that will in real operation satisfy imposed performance requirements cannot be designed. There are well known models in linear SI such as AutoRegressive with eXogeneous input (ARX), AutoRegressive Moving Average with eXogeneous input (ARMAX), Output Error, Box-Jenkins and others, which are widely used in the literature [34] [35]. Furthermore, the polynomial Nonlinear AutoRegressive with eXogeneous Input (NARX) model is considered one of the most desirable techniques in nonlinear SI, because it has proven successful in many applications. In circumstances characterized by complex behavior, CNNs are considered a promising alternative for SI. CNNs form a class of functional representation for nonlinear dynamic systems [14],[28],[33].

A CNN is a biologically inspired computational paradigm which has multiple, interconnected processing elements grouped into layers as linear arrays. Each processing element is characterized by a simple nonlinear operation. The processing power of a CNN in the combination of the specific processing element structure and the network topology. Trained adaptively using a cost criterion, CNNs are believed to be good at modeling of signals and systems. They enable parallel, collective computation and they can implement distributed and sparse memory. A characteristic of a class of CNNs known as perceptrons, includes nonlinear mapping from one set to another, and it leads to their use in nonlinear SI [2]. Various CNN architectures and learning algorithms have been suggested by different researchers and they are used to solve a number of problems of practical importance. One of the well-known learning algorithms developed for the multilayer perceptron (MLP) class of CNNs is the Back-Propagation (BP) algorithm, which is a parameter estimation technique adopting a

gradient descent method. The basic mechanism behind the BP learning rule is the adjustment of the network weights and the bias terms, until the mean-squared-error (MSE) between the output predicted by the MLP and the target reaches a certain acceptable level. Since its development in 1974 and its first wide use in 1986, the BP algorithm has played a significant role in the resurgence of interest in CNNs.

I.1 Brief Notes on Complex Systems

A complex system, or a component of a system, i.e. a subsystem, as used in this study, is characterized by the following features:

- (1) Dynamic behavior, i.e. its measured response exhibits memory,
- (2) Nonlinear behavior, i.e. the linear superposition principle is not applicable,
- (3) Multivariable behavior, i.e. multiple inputs, multiple disturbances, multiple states, and multiple outputs are needed to sufficiently describe its behavior.
- (4) High, frequently infinite, dimensionality, i.e. governed by laws of physics which can be expressed as partial differential equations.
- (5) Poorly understood or not well-understood behavior, i.e. no validated analytical model is available, and,
- (6) Stochastic behavior, i.e. the stochastic component is a significant part of the overall system response.

In this study, systems considered for modeling are assumed to exhibit the aforementioned features. Furthermore, such systems can be categorized into three broad classes:

- (1) Electromechanical systems, including systems with principal behavior governed by rigid body dynamics, i.e. the conservation of momentum and angular momentum,
- (2) Structural systems, systems characterized by flexible dynamics, and,
- (3) Process systems, systems necessitating the laws of thermodynamics and fluid dynamics for adequate description.

On numerous occasions complex systems under study belong to more than one of the aforementioned three broad categories. Nevertheless, in this research study only process systems are investigated. It should be noted that if non-engineering systems are considered, i.e. social or economical systems, then additional classes of systems must be added in the aforementioned categorization.

I.2 Motivation

The objective of this research study is to provide a novel method for designing CNN models for complex process systems, capable of performing accurate MSP and SSP. The motivation behind this work is the lack of effective algorithms available for the aforementioned purposes, in view of the desirable properties of empirical models. For modeling complex process systems, the conventional SI techniques available in the literature, like ARX, ARMAX, NARX, and NARMAX, perform well when tested for SSP, but they fail miserably when tested for MSP.

On the other hand the use physical modeling, has many disadvantages, such as big development cost, numerical complexity, potential inaccuracy to system drifts, inability to incorporate certain poorly understood physical effects, such as wear and

tear etc.. The two-phase flow behavior in a UTSG, is an example where accurate physical modeling is difficult to achieve. The fact that the dynamics of a UTSG are extremely complex, makes it exceedingly difficult to use purely physical models for designing accurate estimators and/or controllers. Physical modeling may also suffer from uncertainties due to an accounted parameters or due to unanticipated plant "drifts". These drifts could be the results of aging, wear and tear, corrosion, material defects, etc. Moreover, physical modeling involves use of large computer programs which require excessively long execution times, and as a result it may not present a practical alternative for on-line application. More importantly, however, on-line (operational) applications require the means to continuously update the available models using sensor information to compensate for the ever present modeling uncertainties and changes in the system behavior. This important feature, which is easily incorporated into empirical models, is exceedingly difficult to incorporate into physical models.

Recently, an emerging approach in nonlinear SI has been to use certain classes of CNNs as empirical models. This has been motivated by the well-known characteristics of CNNs for approximating nonlinear mappings. Thus, in this research a certain class of CNNs, namely the recurrent multilayer perceptron (RMLP), is used as model structure for nonlinear SI. An earlier attempt to use RMLPs to model the UTSG yielded a model which was effective in some MSP transients, though it was very sensitive to the process and sensor noise level in the training and testing data set [16], [42]. The aforementioned model failed when very noisy data were used for training or validation, however, it was effective when the noise level in the data was low enough.

The reason for this relatively poor performance was the fact that the designed predictor was operating effectively in open-loop. Moreover, the designed model could not perform effective MSP during certain severe transients. To do so, the parameters of the RMLP model had to be updated using on-line learning, indicating insufficient predictive performance. The RMLP network trained with the dynamic gradient descent algorithm was used to design this predictive model [16],[45]. However, no attempt was made to use past delayed input(s) and/or output(s) in the RMLP input layer. As a result of some simple numerical experiment performed during that study, there was strong evidence to indicate that a model capable of performing effective MSP without on-line learning and in the presence of high process and sensor noise, can be obtained by incorporating global, as well as local, feedback in the design of the RMLP. By global feedback we mean the use of past (delayed) input(s) and output(s) in the RMLP input layer. Local feedback is inherent in the construction of RMLP.

Empirical modeling, in general, and CNN-based empirical models in particular have many applications, such as forecasting and prediction. Forecasting of future trends is performed by extrapolating models beyond the range over which they were estimated. Forecasts are useful in model validation as well. An accurate forecast may provide information which might lead to the revision of the model that generated the forecast.

Another application of empirical models is the field known lately as control engineering. Model-Based Control (MBC) and Model Predictive Control (MPC) refers to the direct use of an explicit and separately identifiable model for controlling a process. It can be effectively used to control any system provided a suitable, i.e. accurate though not complex system model exist. Empirical models can be used along

with numerous control configurations, where the need for learning specific system dynamics arises.

A third application of empirical models which is gaining a lot of ground throughout industry is the area of condition monitoring and faults diagnosis. Fault-tolerance in dynamic systems is traditionally achieved through the use of hardware redundancy. New approaches have been developed which seek to eliminate some or most of the redundant hardware. These new approaches are based on the idea that two (or more) dissimilar sensors measuring different variables can be used in a comparison scheme to detect any fault in the system [48]. These functionally redundant schemes are basically signal processing techniques employing state estimation and parameter estimation methods, which can be performed in high-speed digital computers using software. Fault detection schemes are designed under the assumption that either the dynamic states of the system under study are known to a certain precision, or it is possible to determine the values of certain physical parameters by on-line identification techniques. CNN-based empirical models appear to be key to the success of these fault detection schemes.

1.3 Biological Neuron vs. Computational Neuron

Perceptron-like CNNs are based on an artificially designed functional neuron. Details regarding this artificial neuron are given in the appendix. The behavior of the functional neuron is quite different than the behavior of a real biological neuron. For comparison purposes a more detailed model of this biological neuron is briefly and qualitatively described in this section.

Biological neurons respond to external stimuli by firing identical pulses over a short period of time. They fire at higher rates when the stimuli are higher. Experimental evidence shows that an absolute refractory period exists immediately after a neuron fires, and that its sensitivity to stimulation grows over time until it is exquisitely sensitive to minute stimuli. One the most prominent biological neuron models that has appeared in the literature describing the aforementioned effects, are those which describe the transmembrane electrical potentials in neurons in terms of their ionic basis. These transmembrane potentials are maintained by differential concentrations of ions and served by ionic currents. One of the earliest models which explains this electrical potentials, and more precisely the resting potential, is the Goldman model [21]. This model was developed on Bernstein's fundamental concepts, on early work by Nernst, and on fundamental physics of ionic media as contributed by Plank and Einstein [36]. The model gives precise, quantitative predictions regarding modulations in transmembrane electrical potentials in neurons, as they are influenced by variations in the constituency of intra and extra fluids and by modulations in the permeabilities of the membrane to the various relevant ions.

Another highly publicized biological neuron model is the Hodgkin-Huxley model [21]. This model describes the transmembrane potential in the course of single-action potentials in term of underlying membrane conductance modulations. It also describes quantitatively the causal mechanisms underlying the generation of action potentials. Another, equally prominent, model is the Hill model which accounts for threshold variations in the process of depolarization and hyperpolarization [21],[54].

The MacGregor model is a combination and an extension of the Hill model, the Hodgkin-Huxley model, and the Kernell & Gutason model [36]. This model provides a quantitatively accurate model of the mechanisms and processes thought to be responsible for fundamental properties of repetitive firing in the triggering sections of neurons, including basic processes of accommodation and adaptation. The main physiological features of the model include the following [36]:

- (1) It produces phasic and tonic cells, including on and off transients, in a predictable manner by varying the appropriate parameters.
- (2) It matches data on adaptive behavior as studied quantitatively in steadily firing mononeurons by Gutason and Kernell.
- (3) It provides a believable basis for the generation of trains of action potentials reflecting ongoing interaction of excitatory and inhibitory synaptic bombardment and current inflow at the triggering section of neurons.

The first manifestation of the approaching impulse is an initial depolarization of the membrane. After an initial $15mV$ depolarization, the rate of depolarization increases. The point at which this change in depolarization rate occurs is called the firing level. Following that, we have the overshoot and a reverse response towards the resting level. When the repolarization is about 70% completed, the rate of repolarization decreases and the curve approaches the resting level more slowly. The sharp rise and rapid fall are the spike potential of the axon, and the slower fall at the end of the process is the after-depolarization. After reaching the previous resting level, the tracing overshoots slightly in the hyperpolarizing direction to form the small, but

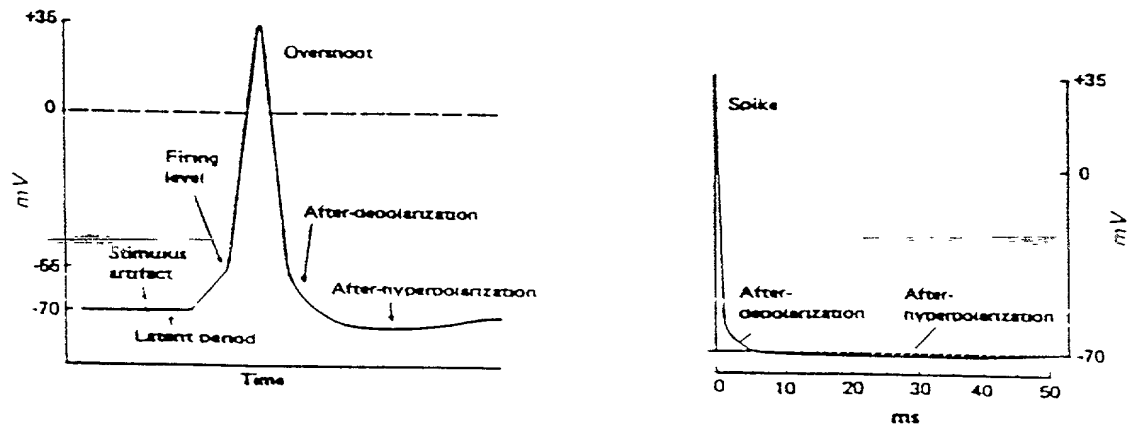


Figure 1. A Theoretical Action Potential of a Biological Neuron (Left), and an Experimental Action Potential of a Mammalian fiber (Right).

prolonged after-hyperpolarization. The whole sequence of potential changes is called the action potential. Figure 1 shows a detailed plot of theoretical action potential, and an experimental diagram of a complete action potential of a large mammalian myelinated fiber, drawn without time or voltage distortion to show proportions of components. The resting potential is $-70mV$. In a preparation such as that in Figure 1, it is possible to experimentally determine the relationship between the strength of the stimulating current and the length of time it must be applied to the nerve to produce a response. Stimuli of extremely short duration will not excite the

axon no matter how intense they might be. With stimuli of longer duration, the threshold intensity is related to the duration of the stimuli. With weak stimuli, a point is reached where no response occurs no matter how long the stimulus is applied [21],[30].

The above brief description clearly shows that biological neurons operates quite differently than the functional neurons used in CNNs. However, the new wave of thinking is to improve the CNN functional neurons based on the observed behavior of biological neurons.

I.4 Literature Review

I.4.1 System Identification

SI can be categorized into two main categories: Linear system identification and nonlinear SI. However, most of the available literature on SI is concentrated in the linear domain. Linear systems are usually idealizations of nonlinear systems encountered in real-world applications. If the system nonlinearities are negligible, then application of linear SI leads to good results. Linear model structures are classified as input-output model structures and state-space model structures.

An input-output model structure is as a rational function with unknown parameters, the coefficients of the numerator and denominator polynomials. An example of such a model in the predictor form, is given by the following equation:

$$\hat{y}(k/k-1) = B_1 u(k-1) + \dots + B_{n_b} u(k-n_b) - A_1 y(k-1) - \dots - A_{n_a} y(k-n_a) \quad (1)$$

where $\hat{y}(k/k-1)$ denotes the predicted value of the system output vector at time-step k given sensed information up to and including time-step $k-1$. Note that a feedforward term can be included in the predictor of equation (1), by adding the $B_0 u(k)$ term to the right-hand-side. However, this term is omitted here. The input signal vector is $u(k)$ while the output signal vector is $y(k)$; for a data set composed of N samples, the time-step k can take values of $1, 2, \dots, N$. Furthermore, n_a and n_b are the number of delays used for the output and input, respectively, and these may be different for the different inputs and outputs of the system. The adjustable parameters of the predictor depicted in equation (1) are the entries of the matrices A and B , for $i = 1, 2, \dots, n_a$ and $j = 1, 2, \dots, n_b$. We call the predictor of equation (1) a multivariable AutoRegressive with eXogenous input (ARX). In the special case where $n_a = 0$, equation (1) is known as the Finite Impulse Response (FIR) predictor. Additional flexibility to equation (1) can be added by including the so-called disturbance dynamics as a Moving Average (MA) process. In predictor form, this results in the following equation:

$$\begin{aligned} \hat{y}(k/k-1) = & B_1 u(k-1) + \dots + B_{n_b} u(k-n_b) - A_1 y(k-1) \\ & + \dots - A_{n_a} y(k-n_a) + C_1 \epsilon(k-1) + \dots + C_{n_c} \epsilon(k-n_c) \end{aligned} \quad (2)$$

where $\epsilon(k-j) \equiv y(k-j) - \hat{y}(k-j/k-j-1)$, is the prediction error (also sometimes called the residual or innovations term). C_i , $i = 1, 2, \dots, n_c$, is the matrix with the disturbance dynamics coefficients, and where all the other terms are as defined for the multivariable ARX predictor. This is a multivariable ARMAX predictor. An ARMA predictor is obtained in the absence of an exogenous input. There exist a

large number of model structures with their corresponding predictors available in the literature [34]. The unknown parameters in these input-output model structures are usually obtained by the least-square method or the maximum likelihood method [61].

For state-space model structures, the relationship between the input, the noise, and the output signals are written as a system of first order differential or difference equations using an auxiliary vector, called the state vector. These models usually give better insight into the physical mechanisms of a process, because they are derived for the physical laws governing the process. There are, however, state-space representations in which the states are of empirical nature. These are called empirical state-space structures, though they are rarely used in practice because of the lack of robust parameter estimation algorithms for them. State-space model structures usually lead to the following predictor representation; also called the innovations representations:

$$\hat{x}(k+1/k) = A(\Theta)\hat{x}(k/k) + B(\Theta)u(k) + K(\Theta)\epsilon(k), \quad (3)$$

$$\hat{y}(k/k-1) = C(\Theta)\hat{x}(k/k-1). \quad (4)$$

Where, $\epsilon(k)$ is the innovations term previously defined and where $\hat{x}(\cdot) \in R^n$ is the system state vector. $A(\Theta)$, $B(\Theta)$, $K(\Theta)$, and $C(\Theta)$ are matrices of appropriate dimensions, and Θ is a vector of parameters that typically correspond to the unknown values of the physical coefficients, material constants, or other non-physical parameters. The modeling is usually carried out in terms of state variables, $x(k)$, which have physical significance, and measured outputs which are linear combinations of these

states. The aforementioned input-output and state-space model structures are said to belong to the family of parametric models. There are other classes of non-parametric models, such as spectral estimators, which are not discussed here [34].

The literature for applied and computationally oriented nonlinear SI is quite scarce, though there have been numerous theoretical studies dating back to the early 1900s. Traditionally, functional series method, such as the Volterra and Wiener series, have been used for the identification of nonlinear systems [3], [19], [23]. It is well known that these structures can describe a number of nonlinear systems, however, they are not very convenient for practical use. Billings and his colleagues were the first to report the polynomial NARX model structure [5], [10], [11], [12]. The polynomial Nonlinear AutoRegressive and Moving Average with eXogeneous input (NARMAX) nonlinear model structure was also introduced by these authors. They have worked extensively in the area of input-output nonlinear SI using a number of methods. The polynomial NARX appears to be one of the most promising structures for representing nonlinear systems. Later in this document both NARX and NARMAX model structures are presented in more detail.

I.4.2 Computational Neural Networks

A CNN is a biologically inspired computational paradigm which has multiple interconnected processing elements grouped into layers as linear arrays. Each processing element is characterized by a simple nonlinear operator. During the past ten years there has been an explosive growth in pure and applied research related to CNNs. Investigators from across the spectrum including neurobiologist, neurophysiologist,

psychologists, mathematicians, computer scientists and engineers have been attracted to this field. Neurobiologist and neurophysiologist are interested in CNNs as a way to understand the physiological functioning of the human neural system, while psychologists study brain functions at the cognitive and behavioral level and they are interested in using CNN-based techniques to create detailed models of human behavior. Inspired by the ability of humans and animals to easily perform such tasks as processing sensory information and interacting with uncertain environment, engineers are attempting to build machines which possess similar information processing capabilities. All involved are excited about the opportunity for cooperative research among scholars in different fields, and the prospects of drawing ideas and perspectives from so many diverse disciplines. It is believed that a meaningful theoretical integration of all this knowledge will become a reality within the next few decades [24], [36],[53].

The processing power of a CNN is a combination of its specific processing element structure and its network topology. A CNN is constructed using functional neurons described in the appendix of this document. Trained by adaptation using a cost criterion, CNNs are believed to be good at interpolation and extrapolation. The path between the nodes (or the counterparts of the biological dendrites) are modeled by the CNN links. The connections between neurons in a network fundamentally determine the behavior of the network and how that behavior can change with time. For this reason, the field today known as neural networks was originally called *Connectionism*. Connectionism was born during the middle decades of this century. In the 1940s, Warren McCulloch and Walter Pitts explored the computational capabilities

of networks made-up of model neurons with a very simple design. A McCulloch-Pitts model neuron fires if the sum of its excitatory inputs exceeds its threshold, so long as it receives no inhibitory input. Networks of this type seemed appropriate for modeling not only symbolic logic, but also perception and behavior. At the same time during the 1940s, Wiener laid the foundations of the field known as cybernetics, or the control and communications in man and machine, which was a prelude to the field of modern artificial intelligence [67]. In the 1950s, Rosenblatt [53] introduced a mathematical analysis of the behavior of a class of network models called perceptrons. The perceptron exhibited some learning and generalization capabilities. After going through many other stages including a phase of significant lack in progress, CNNs or artificial neural networks (ANNs) regained momentum through the work of Hopfield, Rumelhart, Williams, and Werbos. The latter three were instrumental in advancing the so-called Backpropagation (BP) algorithm throughout the later half of the 1980s.

Many CNN architectures and learning algorithms have been suggested by different researchers and they are used to solve a number of problems of practical importance. One of the most promising CNN model structures is the Feedforward Multilayer Perceptron (FMLP) neural network [57]. FMLP networks are characterized by sets of nonlinear algebraic equations, one for each node of the network. The BP learning algorithm, which is a parameter estimation technique used in training FMLPs with a gradient descent method, has played a significant role in the resurgence of interest in CNNs. The RMLP is another promising CNN architecture, which is more sophisticated than the FLMP. For complex process system such as the ones treated in

this research, the RMLP is by far a more appropriate architecture to adopt than the FMLP.

Several techniques and algorithms have been developed for training various forms of recurrent neural networks (networks with feedback and cross-talk connections). Williams and Zipser [68], and Williams and Peng [69] have developed an algorithm for fully connected recurrent networks, the so-called dynamic backpropagation. Narendra and Parthasarathy [39] have discussed the dynamic backpropagation learning algorithm, and its application for the optimization of FMLP neural network parameters. They emphasized the diagrammatic representation of the system which generates the gradient of the performance function. Additionally, Werbos[64], Pineda [50],[51], and Chong [16] have proposed algorithms for training recurrent networks.

CNNs have scored many successes in many areas of human life. One the most striking example is a system called NETTALK [55]. The task of this system is to learn how to pronounce English words. The input to the CNN is English text, and the output of the network is fed to a machine that can produce voice sounds. The ability of the CNN to extract features of English speech was quite remarkable, and naturally it led to the conversion of speech to written text, a field known as pattern recognition. Currently, vast amounts of resources are invested by the United States, countries of Western Europe, and Japan for more research on CNNs.

1.5 Research Contributions

The contribution of this research study is to develop a method for modeling complex process systems using the RMLP. The developed CNN model should be capable

of performing accurate SSP and MSP. A CNN model capable of MSP has many applications in the areas of forecasting and prediction, control engineering, and condition monitoring and fault diagnosis.

In particular, the following two specific contributions have been made as a result of the research described in this dissertation:

- (1) A new dynamic learning algorithm is developed, which, in addition to the local feedback present in the RMLP uses Global Feedback (GF). Additionally, an approach is proposed for designing an RMLP model for good MSP performance.
- (2) The effectiveness of the developed learning algorithm is demonstrated by modeling a U-Tube-Steam-Generator (UTSG), a highly complex process system which is characterized by poorly understood two-phase flow effects, as well as reverse and unstable open-loop dynamics. The performance of the developed CNN-based model is extensively tested.

I.6 Organization of the Dissertation

In chapter II, an overview of SI is presented, and the different applications of SI in the areas of forecasting and prediction, control engineering, and condition monitoring and fault diagnosis are briefly described.

Chapter III discusses the Recurrent Multilayer Perceptron (RMLP) network with the different learning algorithms used in training it. The GF technique and the Teacher Forcing (TF) technique are discussed. Also, issues of network complexity are addressed in this chapter.

In chapter IV, the polynomial NARX, a powerful technique in conventional non-linear SI, is first discussed in detail. Then a two-input two-output (2I2O) system is presented as a case study, comparing the traditional methods of SI with CNN methods.

In chapter V, a UTSG empirical neural network model is developed. The GF technique, the TF technique, and the polynomial NARX technique are all investigated and compared. The impact of noise on training and validation is addressed, and an extensive model validation study is also performed. A summary and conclusions from this research are given in chapter VI. A detailed description of the perceptron-like functional neuron used throughout this study is given in the appendix.

CHAPTER II

SYSTEM IDENTIFICATION AND ITS APPLICATIONS

II.1 Introduction

When interacting with a system (static or dynamic), we need some concept of how its variables relate to each other. With this broad definition, we shall call such an assumed relationship among observed signals a *model* of the system. Models may come in various shapes and be phrased with varying degrees of mathematical formalism. For certain systems it is appropriate to describe their properties using numerical tables and/or plots, and we call such descriptions *graphical models*. Linear systems, for example, can be uniquely described by their impulse or step responses or by their transfer functions. Graphical representation of these systems are widely used for various design purposes. For more advanced applications, however, it may be necessary to use models that describe the relationships among the system variables in terms of mathematical expressions like difference or differential equations. For the case of a U-Tube Steam Generator (UTSG), for example, such differential equations are based on the conservation of mass, momentum, and energy. We call such models *physical models*. There exists another type of models called *empirical models*. These models are developed using observed data from the system under consideration. Neural network models are considered to be special classes of empirical models. Both physical models and empirical models are considered mathematical models, as opposed to graphical models.

The use of the mathematical models is inherent in all fields of engineering and science, including social, natural and management sciences. A major part of current engineering field practice deals with the development and utilization of accurate and useful mathematical models. The model used in a computer simulation of a system is itself a computer program. For complex systems, this program may be composed of many interconnected subroutines and look-up tables. The term software model is used for such computerized descriptions.

System identification (SI) deals with the problem of building empirical models of systems based on data observed from the system. Many theories have been developed in this area, and they are well understood and known to perform successfully in practical applications. In recent years, the importance of SI has increased in many fields such as economics, medicine, and process control. Especially in engineering applications, model building has become one of the most important and time consuming aspects of system design and operation. Design of models is necessary for understanding the dynamics of a system, and for the mathematical analysis of systems. Without an adequate model of a system to be controlled, for example, a good controller that will satisfy the desired performance requirements cannot be obtained.

The remainder of this chapter is organized as follows: The next section briefly describes the SI procedures. Section II.3 through section II.5 present the different applications of SI in the areas of forecasting and prediction, control engineering, and condition monitoring and fault diagnosis, respectively. Section II.6 presents an evaluation of the conventional algorithms in SI and their limitations.

II.2 System Identification Procedures

SI is comprised of several sequential stages [34], [59], which are shown in Figure 2 in terms of a flowchart. These can be briefly described as follows:

- (1) Experiment design and data collection: The purpose of the experiment is to generate some input-output data sets. These data sets are recorded during a specifically designed identification experiment, where the user may determine which signals to measure, when to measure them, and which input signals to choose. The type of input signals, the range of input signal amplitudes, the frequency ranges and the sampling rates are factors to be considered. The input signals must satisfy certain conditions. A minimum requirement is that the dynamics of the model have to be excited persistently by the input signal over the measurement period. This means that during the experiment the input signal must be sufficiently rich to excite all of the dynamic modes of interest. The objective of the experiment design is thus to make these choices appropriately, so that the data become maximally informative. Apriori knowledge about the system under study is a key factor in the experiment design and data collection, as well as all other SI stages. Once the data have been collected, they should be evaluated using graphical inspection, as well as some correlation tests. Correlation tests are well-known statistical tools that can evaluate the goodness of the collected data. If the collected data fail to pass these tests, the experiment must be redesigned and new data must be collected. The process is repeated until the data are fit for use in SI.

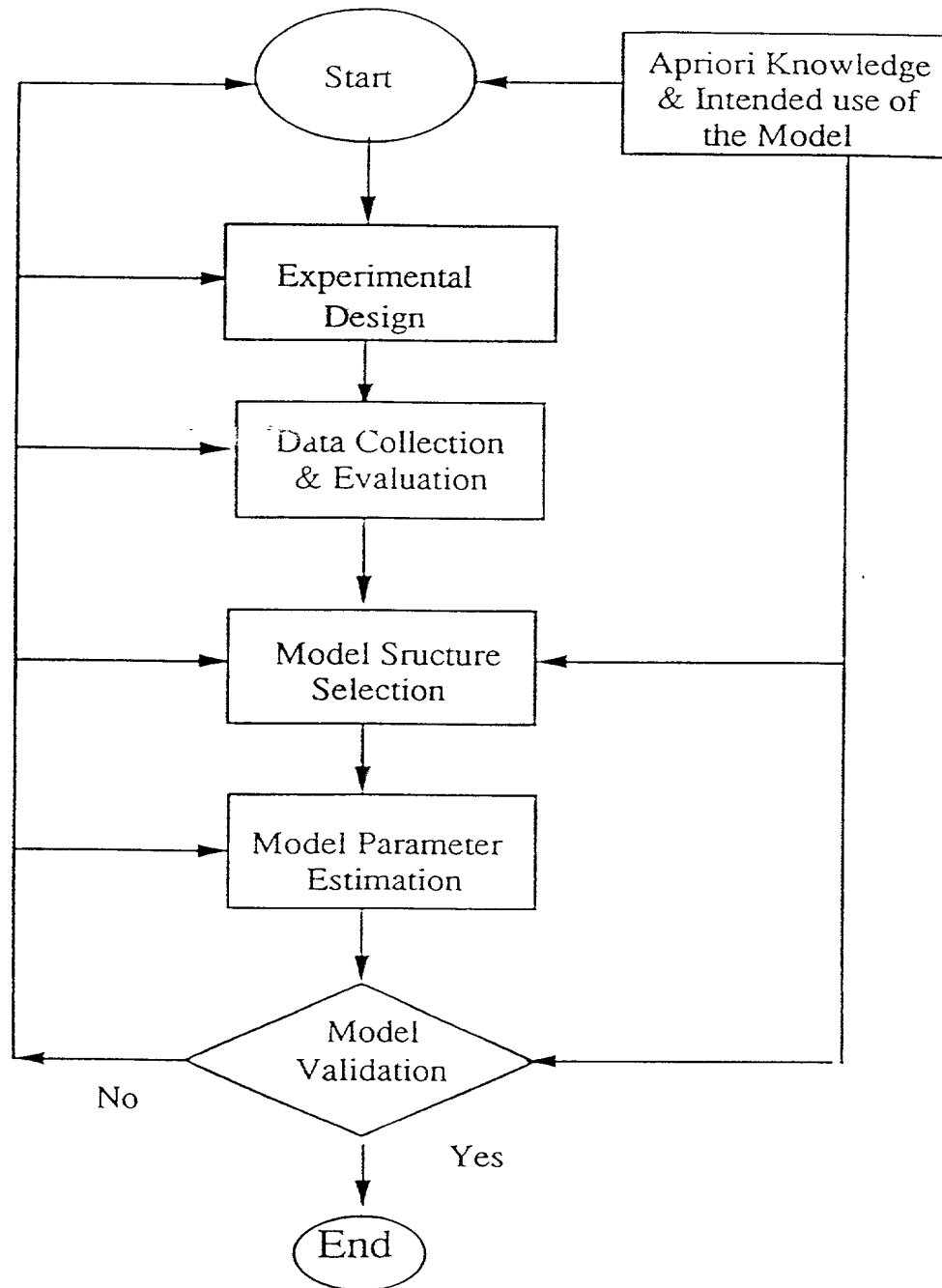


Figure 2. Stages of the System Identification Process.

- (2) Model structure selection: A set of candidate models is selected. This is the most important choice in SI procedures. At this stage, apriori knowledge, engineering intuition, and insight have to be combined with formal properties of the model. As stated earlier, model structures are classified into two classes, input-output model structures and state-space model structures. The choice of a model structure depends heavily upon the system under study.
- (3) Model parameter estimation: This stage involves model parameter estimation using techniques like the least-squares method, or the maximum likelihood method, a choice which critically depends upon the selected model structure. These methods are published in numerous books which deal with the fields of SI and dynamic system modeling [18],[34],[35].
- (4) Model validation: Having gone through the preceding three choices, we have at least implicitly arrived at a particular model: This is the one that best describes the data according to the chosen criterion. It remains to test whether this model is “good enough”, that is, whether it is valid for the purpose it was designed for. Such tests are known as *model validation*. Residual analysis is a well-known method in the field of SI and statistical analysis [8],[9]. This method uses the residuals, which are the differences between the predicted and the observed outputs. The intent is that if the residuals are white and independent of the model input, then the estimated model is good. Statistical tests for the autocorrelation function of residuals and the cross-correlation function between residuals and input are applied in the residual analysis stage. The autocorrelation determines the “whiteness” of the residuals. Furthermore, it is known from statistical analysis that if the sequence of residuals

is white, then they would be asymptotically chi-square (χ^2) distributed. Hines [25] gives more information on chi-square (χ^2) distribution. Another way to check for model validity, is to test the model using a new data set not used in the process of designing the model. This latter approach will be primarily adopted for the purpose of this research, because of the nonlinear nature of the models developed.

Other approaches will be disregarded, for reasons to be explained in more details

later in this study.

II.3 Applications in Forecasting and Prediction

A forecast is a quantitative estimate (or set of estimates) about the *likelihood* of future events based on past and current information as well as on assumptions regarding future events. This information is usually embodied in the form of a model. By extrapolating the models beyond the period or range over which they were estimated, we can make forecasts about future events. The term forecasting is often thought to apply solely to problems in which a future event is predicted. The information provided by the forecasting process can be used in several ways. Frequently forecasts are used for public and private policy making purposes. For example, a forecast of a high rate of inflation based on the assumption of a large budget deficit may lead policy makers to alter their budget plans, or a forecast of an increased world demand for crude oil may lead shipbuilders to invest in new supertankers. Forecasts are also useful as model validation tests. A forecast which is found to be "off target" when the actual observation become available, provides information which may lead to the

revision of the model that generated the forecast. Two types of forecasts can be useful. *Point forecasts* predict a single number in each forecast period, while *interval forecasts* indicate an interval in which there is a likelihood the realized forecast value will lie. Both types of forecasts are encountered in engineering, depending upon the forecast uncertainty involved in the forecasting horizon.

Time series and econometric analysis are classes of conventional techniques used in the field of forecasting. Time series include the use of the Kalman filter, the Box-Jenkins model, the ARMA model, and other techniques. Econometric approaches include linear, piecewise linear or simple nonlinear functions as the basic functional elements for which the proper coefficients are to be found using techniques such as least-square [40]. Computational Neural Networks (CNNs) are believed to be a much more powerful technique than the aforementioned conventional methods [40], [49]. One of the potential consequences of this research study would be to apply the capabilities of CNNs in the field of forecasting, especially in the area of long-term forecasting. This can be accomplished by accurate identification of the system under consideration. Once the identification of the system is completed using an empirical model based on CNNs, long-term forecasting can be performed.

II.4 Applications in Control Engineering

A control problem can be defined by the following sentence: It is an on-line update of the manipulated variables to satisfy multiple and changing performance criteria in the face of changing system dynamic characteristics. The whole spectrum of control methodologies in use today is faced with the solution of this problem. The

difference between these methodologies lies in the particular assumptions and compromises made in the mathematical formulation of the performance criteria and the selection of a system representation. These are made primarily to simplify the mathematical problem so that its solution fits an existing analytical and/or computational approach. One of the most crucial compromises made in control engineering is to ignore constraints in the formulation of the problem. These compromises can deny the control system its achievable performance. Model Predictive Control (MPC) techniques provide a methodology to handle constraints in a systematic way during the design and implementation of the controller [22]. MPC refers to the direct use of an explicit and separately identifiable model for controlling a process. It can be effectively used to control a process system provided that a suitable system model exists. CNNs can be used within the MPC framework, as well as other frameworks, where the need for learning specific dynamics arises. Werbos has summarized the five basic designs where CNN can be effective in control engineering, as follows [66]:

- (1) *Supervised control*: It can be used to copy a computer-based controller, for a smaller and faster copy in real-world application. CNN is an efficient way to perform this task. The neural network is taught the mapping from sensors inputs to the desired control actions, based on a history of actions provided by a human expert or a computer program.
- (2) *Direct inverse control*: In this technique, the neural networks directly learn the mapping from desired trajectories (e.g., of a robot arm) to the control signals which yield these trajectories (e.g., joint angle). This technique is successful provided there is a unique mapping from the trajectory coordinates to the actuator

coordinates. Direct inverse control has some drawback, because it does not have the ability to exploit extra actuators to achieve maximally smooth, or energy-saving motion [66]. Both direct inverse control, as well as supervised control, are straightforward applications of supervised learning, and they do not involve any planning or optimization in achieving the control objectives.

- (3) *Neural-adaptive control*: Instead of the linear mappings used in conventional adaptive control architectures, neural networks are used for the adaptation of the controller parameters so as to keep the actual behavior of the system consistent with the control specifications. The adaptive neurocontrollers are based on the principles of SI and its use for control. Research has been focused on Model-Reference-Adaptive Control (MRAC) and Self-Tuning Regulators (STR) as implementations of neuro-adaptive control [63].
- (4) *Backpropagation of utility*: This is a direct model-based utility maximization approach, where the gradients of the utility function in future time with respect to current actions are calculated. The derivatives so obtained are used to tune the neural network outputs which are the control actions applied to the system. The limitations of this approach lie in the fact that a differentiable model of the process is required. If the model involves dynamics, then off-line adaptation of the action network is required [47]. This technique is not efficient because it cannot account for noise and cannot provide real-time learning for very large problems.
- (5) *Adaptive critic methods*: This method attempts to approximate dynamic programming, which is the only efficient method for finding an optimal strategy of action

over time in a noisy, nonlinear environment. This technique is consistent with real-time learning and exhibits good performance for considerable noise levels [66].

The techniques described above show that CNNs have a significant potential in the field of control engineering. The fact that CNNs can be used successfully as system identifiers makes them ideal candidates for use in neurocontrol. As stated earlier, the objective of this research study is to arrive at a method which can be used to design CNN-based models capable of learning all or the most important dynamics of a complex process system. As a result, it is believed that the outcomes of this research study are applicable for enhancements in the field of neurocontrol.

II.5 Applications in Condition Monitoring and Fault Diagnosis

The complex systems widely used in various industries consist of hundreds of inter-dependent working parts, which are individually subject to malfunctions and failures. A failure in these systems can present a significant economic loss and/or hazard to personnel. Therefore, it is necessary to provide a *monitoring scheme* which detects a fault as it occurs. This monitoring scheme must be able to identify the malfunction of a faulty component by substituting a configuration of redundant elements so that the system continues to operate safely in a satisfactory manner. This field is known as *fault detection and isolation* (FDI), or simply fault diagnosis [48]. Fault diagnosis algorithms employed in FDI systems are usually implemented on a digital computer. These algorithms are essentially signal processing techniques combined with logical switching functions. The overall system reliability depends on the reliability of both

the physical components of the system and the computer which is assisting or monitoring its behavior. Computer malfunctions or failures are usually considered less likely, and as a result they are not considered part of FDI.

Fault-tolerance in dynamic systems is traditionally achieved through the use of hardware redundancy, where protection against localized damage is provided by spatially distributing repeated hardware elements around the system. This approach can be simple and usually straightforward to apply. As a result, it is widely used in the control of aircraft, space vehicles, and in certain process plants, like nuclear power plants and plants handling dangerous chemicals. However, hardware redundancy comes with extra cost, software, and additional space to accommodate the equipment. This issue becomes very important, especially in the aerospace industry. To overcome these problems and to improve overall system reliability, new approaches have been developed which seek to eliminate all or most of the redundant hardware.

These new approaches are based on the idea that two (or more) *dissimilar* sensors measuring different variables can be used in a comparison scheme to detect any fault in the system. The logic behind this idea is that although the sensors are dissimilar in terms of the signals each sensor is measuring, they are all driven by the same dynamic states of the system and they are therefore *functionally* related [48]. These functionally redundant schemes are basically signal processing techniques employing state estimation, parameter estimation, and statistical decision theory, all of which can be performed in high-speed digital computers using software. The ingredients used by the software are the sensors signals. The FDI schemes are therefore designed under the assumption that either the dynamic states of the system under study are

known to a certain precision, or it is possible to determine the values of certain physical parameters by on-line identification techniques. On-line means that the FDI scheme analyzes and obtains information regarding the behavior the system while the system is in operation.

A generic architecture for condition monitoring and fault diagnosis is shown in Figure 3, where three distinct phases can be observed [27],[48]:

- (1) *Data Processing*: This phase consists of observing signals of the system which may carry a fault along with a system model which is fault-free. The fault-free system model is either a physical model or an empirical one, assumed to be available.
- (2) *Condition Monitoring*: Following the information processing in phase one, state or parameter estimation is performed, and residuals are generated.
- (3) *Fault Diagnosis*: Any significant discrepancy between the variables of the observed system model and the fault-free system model, are considered faults. Following the detection of a fault, its features are classified to determine the fault type, location, size, and the cause of the fault. The final stage may consist of corrective action against the fault.

Fault diagnosis techniques can be also applied to monitor plant stability. In addition to the integrity of plant components, such as sensors and controllers, attention must be paid to the dynamic characteristics of the plant itself due to its own ultimate importance. The possibility of on-line monitoring of system stability margin has been suggested from the early stages of nuclear power plant development, but the actual implementation of these on-line techniques became feasible only after recent development in computer technology. The problem of on-line instability monitoring has been

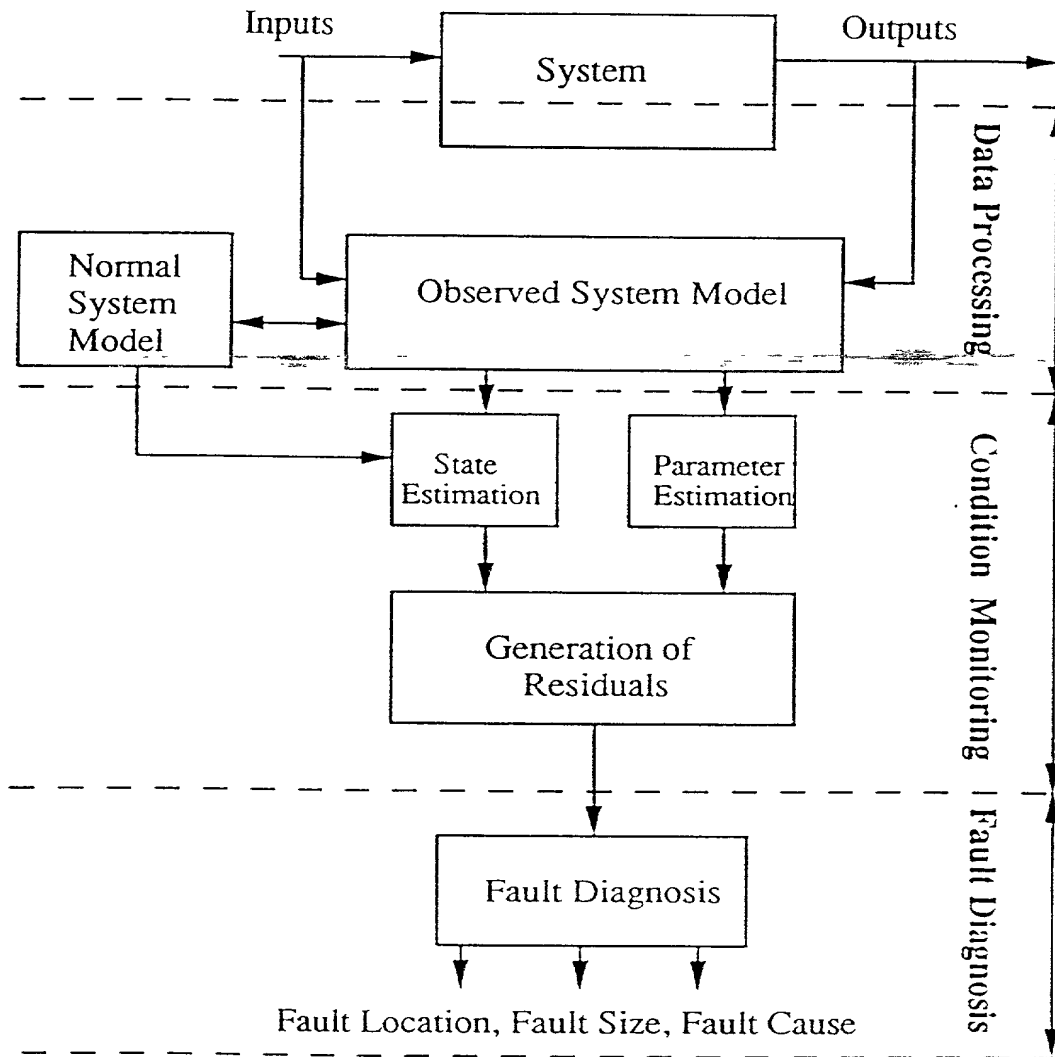


Figure 3. Generic Architecture for Condition Monitoring and Fault Diagnosis

considered mostly in Boiling Water Reactor (BWR) plants because of their inherent instabilities: i.e. tight coupling between nuclear and thermal-hydraulic effects in the reactor core. Two types of instabilities are considered possible in BWR: these are *reactivity* instability and *channel* instability. The former arises due to the interaction of the neutronic and the thermal-hydraulic processes, and is related to the stability of neutron flux response to a perturbation in reactivity, such as control rod action and coolant flow rate change. The latter is caused by the two-phase flow dynamics of a heated coolant channel. Early detection of these instabilities is highly desirable because this contributes to the reduction of the number of scrams ; i.e. forced outages in nuclear power plants.

The material presented in this section indicates that for fault diagnosis to become practically realized, an empirical model with on-line implementation capability should be available. This brings us to the central theme of this research study, which has the objective designing a CNN-based model capable of capturing the dynamics of complex process systems, and which can be utilized during system operation.

II.6 Limitations of Conventional Identification Algorithms

Most of the research on SI has concentrated in the linear domain. Ljung and Gunnarsson[32] have presented a survey of basic algorithms for tracking linear, time-varying systems. Horak [26] presented a method for identifying the errors of a linear, time-invariant model of a dynamic system based on system inputs and outputs collected during an experiment. The method is based on linear programming techniques

which guarantee that the identified errors are the smallest possible which can reproduce the experimental data.

Chen and Billings [11] reviewed the modeling of several nonlinear systems under the framework of a general representation known as the polynomial NARMAX model structure. Furthermore, they have shown that the polynomial NARMAX model structure can be a general and natural representation for a large class of nonlinear systems [11]. Their work represents a significant contribution in the applied aspects of nonlinear SI. Additionally, Chen and Billings [13] derived a recursive prediction error parameter estimation algorithm for the polynomial NARMAX model, and Billings extended an orthogonal least square estimator, originally derived for Single-Input Single-Output (SISO) systems, to Multi-Input Multi-Output (MIMO) nonlinear systems [4],[5].

All the aforementioned techniques give satisfactory results for single-step-ahead prediction (SSP). SSP means that the model can predict the system output at a certain time-step given sensed information up to and including the previous time-step. In case the sensors fail to give an accurate reading of the system output, a model utilized as a one-step-ahead predictor will fail to predict the correct system response. This issue is of great importance especially in the area of adaptive control, where the sensed system output is the only means by which the system can be controlled. Note that the ability of the conventional techniques to provide a model capable of performing accurate SSP does not mean that the model has captured the deterministic dynamics of the system under study. Only a model capable of performing accurate

multi-step-ahead-prediction (MSP) can be classified as such. It has been our experience that the conventional algorithms are not good at MSP in complex process systems. This will be demonstrated in the case studies presented in the following chapters.

The ultimate objective of this research study is to design empirical model, which capture the dynamics of complex systems, and which can be utilized under the worst possible operating condition, such as sensors' failures. Thus, it is desired that the developed empirical models perform accurate MSP. This means that the model should accurately predict future system outputs within a finite time horizon. A model capable of such a task is a model which has captured all or most of the deterministic dynamics of the system under study. Moreover, the model should be accurate in performing MSP even in the presence of high process and sensor noise associated with the system, a frequently encountered circumstance for most industrial systems. Such empirical models are different than physical models, because they have the ability to perform on-line learning. On-line learning can continuously update an empirical model to accommodate drifting dynamics during the lifetime of a system. These drifts in system dynamics might be due to aging, failures, erosion, and other factors, and they are important to incorporate in on-line models for effective predictor, control, and condition monitoring and fault diagnosis.

CHAPTER III

THE RECURRENT MULTILAYER PERCEPTRON NETWORK
AND ITS USE AS A NONLINEAR PREDICTOR

III.1 Introduction

This chapter discusses the Recurrent Multilayer Perceptron (RMLP) network, a special class of recurrent Computational Neural Networks (CNNs), commonly known in the literature as recurrent Artificial Neural Networks (ANNs). The use of the RMLP as a nonlinear predictor is discussed and two learning algorithms are derived.

A CNN is a biologically inspired computational paradigm which has multiple interconnected processing elements grouped into layers as linear arrays. Each processing element is characterized by a simple nonlinear operation. The details of a simple CNN processing element are given in the Appendix. The processing power of a CNN is a combination of the specific processing element structure and the network topology. Trained by iteratively using a cost criterion, CNNs are believed to be good at extrapolation and interpolation. They are capable of parallel, collective computation and they exhibit distributed and sparse memory. The path between the nodes (or the counterparts of the biological dendrites) are modeled by the CNN links. The inputs to the nodes are the outputs of other nodes, weighted by the efficacy of the transmission links between the source and destination nodes. The nodes algebraically sum these weighted signals (activation signals) they receive, and produce a net output. The resulting net output is then passed through a discriminatory function, and the

output of the node is thus obtained. The discriminatory function, or the squashing function, can be a sigmoid function or a hyperbolic tangent function.

Many CNN architectures and learning algorithms have been suggested by different researchers and they are used to solve a number of problems of practical importance. One of the most promising CNN model structures is the Feedforward Multilayer Perceptron (FMLP) neural network [57]. FMLP networks are characterized by sets of nonlinear algebraic equations, one for each node of the network. The backpropagation (BP) algorithm, which is a parameter estimation technique adopting a gradient descent method, is widely used for training FMLPs and it has played a significant role in the resurgence of interest in CNNs [66]. The RMLP is another promising CNN architecture, recurrent in nature, which is more complex than the FMLP. As demonstrated in this and other studies, for complex process systems, like the one considered in this research, the RMLP is by far a more appropriate architecture to adopt than the FMLP.

Several techniques and algorithms have been developed for training various forms of recurrent neural networks, i.e. networks with feedback and cross-talk connections. Williams and Zipser [68], and Williams and Peng [69] have developed an algorithm for fully connected recurrent networks, the so-called dynamic BP. The weights are updated by dynamically adjusting the gradients based on the inputs and the target outputs. However, the feedback connections are global, that is only the network output is utilized as an input for the network itself. Narendra and Parthasarathy [39] have discussed the dynamic BP learning algorithm, and its application to the optimization of FMLP neural network parameters in a straightforward manner. They

emphasized the diagrammatic representation of the system which generates the gradient of the performance function. Additionally, Werbos[64] and Pineda [50],[51] have proposed algorithms for training recurrent networks. Furthermore, Principe [52] has shown how the Gamma model can be used to add memory to CNNs, however, this technique has been applied to speech recognition problem only.

This chapter is organized as follows: The next section gives a brief discussion of the FMLP and the RMLP architectures which are used in nonlinear system identification (SI). Sections III.3 and III.4 present two learning algorithms for the RMLP, the previously reported dynamic gradient descent learning algorithm with Teacher Forcing (TF), and the newly developed dynamic gradient descent learning algorithm with global feedback (GF), respectively. The last section covers some issues related to network architecture design. The chapter ends with a summary.

III.2 Computational Neural Networks Architectures

In this section the two CNNs architectures utilized in this research are discussed in some detail.

III.2.1 The Feedforward Multilayer Perceptron

The CNN shown in Figure 4 is called an FMLP. From a SI stand point, it can be considered a nonlinear input-output model structure. The network is composed of an input layer, a series of hidden layers and an output layer. In this network, the signals from each node are transmitted to all the nodes in the next layer, where only the hidden layers have a sigmoid-type discriminatory function. The input and the

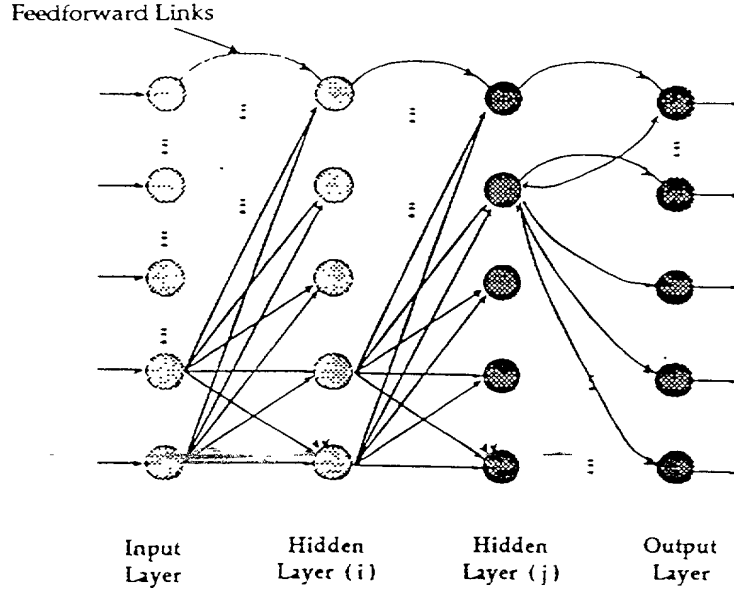


Figure 4. The Feedforward Multilayer Perceptron Network.

output layers have linear discriminatory functions and no biases [41]. FMLPs without appropriate signals in the input layer are good at approximating static nonlinearities, i.e. memoryless nonlinear functions. Training of the network adjusts the active range of the discriminatory function, such that mostly the linear range is used for achieving a piecewise linear approximation [11].

The processing elements of an FMLP network are governed by the following equations:

$$x_{[l,i]}(k) = F_l \left(\sum_{j=1}^{N(l-1)} w_{[l-1,j][l,i]} x_{[l-1,j]}(k) - b_{[l,i]} \right). \quad (5)$$

for $i = 1, \dots, N(l)$ (the node index), and $l = 1, \dots, \mathcal{L}$ (the layer index), where $x_{[l,i]}(k)$ is the i th node output of the l th layer for sample k , $w_{[l-1,j][l,i]}$ is the weight (the adjustable parameter) connecting the j th node of the $(l-1)$ th layer to the i th node of the l th layer, $b_{[l,i]}$ is the bias (also an adjustable parameter) of the i th node in the l th layer, and $F_{[l]}(\cdot)$ is the discriminatory function of the l th layer. No analysis issues related to the FMLP are presented here, though there have been numerous authors who have proven the ability of FMLP's to approximate almost any nonlinear function [17].

III.2.2 The Recurrent Multilayer Perceptron

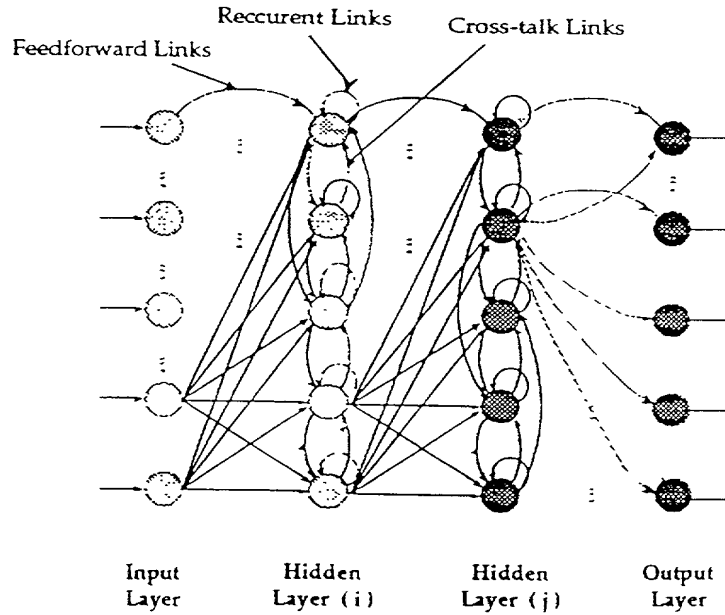


Figure 5. The Recurrent Multilayer Perceptron Network.

Another type of a CNN is the Recurrent Multilayer Perceptron (RMLP) shown in Figure 5. The main distinction between the RMLP and the FMLP is the existence of recurrent and cross-talk (delayed) links in its hidden layers. These links feed the output of the node back to itself or to other nodes (cross-talk) with one time-delay. These time-delayed links thus impart memory to the node, and past information can be processed along with current node and, in general, network inputs. The output of each node of the RMLP is a function of its internal node state its inputs, and indirectly the network inputs, and hence it is believed that the dynamics of a system can be approximated by the RMLP. Because of its inherent memory capabilities, an RMLP can be considered a nonlinear empirical state-space model structure, as demonstrated in the following section, and it has been demonstrated to be more effective than an FMLP model structure in modeling complex nonlinear systems [16].

The equations describing the i th node at the l th layer of the RMLP network are given by the following difference equations:

$$z_{[l,i]}(k) = \sum_{j=1}^{N(l)} w_{[l,j][l,i]} x_{[l,j]}(k-1) - \sum_{j=1}^{N(l-i)} w_{[l-1,j][l,i]} x_{[l-1,j]}(k) + b_{[l,i]}, \quad (6)$$

and

$$x_{[l,i]}(k) = F_{[l]}(z_{[l,i]}(k)), \quad (7)$$

where $z_{[l,i]}(k)$ represents the internal state variable of the i th node at the l th layer for example k ; $x_{[l,i]}(k)$ is the i th node output of the l th layer for example k , and $b_{[l,i]}$ is the bias of the node ; $w_{[l,j][l',i]}$ is the weight associated with the link between the the j th node of the l th layer to the i th node of the l' th layer. Furthermore, k represents

the discrete-time at which the network outputs are computed, with the node index $i = 1, \dots, N(l)$ and layer index $l = 1, \dots, \mathcal{L}$, and with the $F_{[l]}(\cdot)$ for the input and output layers ($l = 1$ and $l = \mathcal{L}$) being linear. The term $b_{[l,i]}$ provides the bias for each node, defining the region where each node is "active". Throughout this study, hyperbolic tangent discriminatory functions have been used in the hidden layer nodes. Note that the RMLP network described by equations (6) and (7) can be reduced to a purely feedforward network, i.e. an FMLP, by simply setting the intra-layer weights to zero, i.e. by eliminating the first term in the right hand side of equation (6).

III.3 Single-Step-Ahead and Multi-Step-Ahead Prediction

Strictly speaking Multi-Step-Ahead Prediction (MSP) is the estimation of the system output at some time-step $k + p$ based on output information up to time-step $(k - 1)$, i.e. calculation of $\hat{y}(k + p/k - 1)$. If $p = 0$, then the prediction is called single-step-ahead prediction (SSP). Figure 6 demonstrate the concept of SSP and MSP.

Theoretically speaking, in order to perform MSP one would have to directly relate $\hat{y}(k + p/k - 1)$ with $y(k - 1), y(k - 2), \dots$ etc. Nevertheless, this is hardly ever done, and in practice MSP is performed recursively, by relating $\hat{y}(k + p/k - 1)$ with $\hat{y}(k + p - 1/k - 1), \hat{y}(k + p - 2/k - 1), \dots$ etc. In other words, in order to perform a $(p + 1)$ -step-ahead prediction, one would perform $(p + 1)$ SSP recursively, i.e. by utilizing the prediction in one time-step to perform the prediction in the next time-step. In fact, in the context of an ARMA process, and under the Gaussian hypothesis, it can be

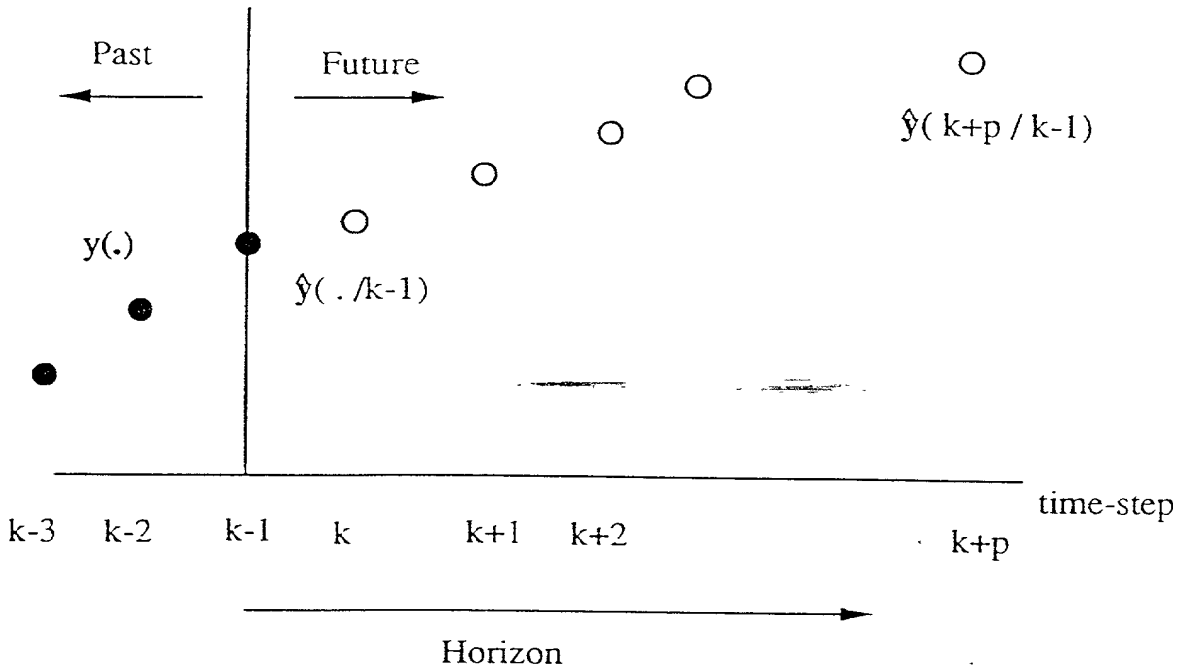


Figure 6. Multi-Step-Ahead Predict Schematic (p+1)-Step Horizon.

shown that the optimal mean-square MSP error can be obtained by such a recursive procedure [59]. Furthermore, it can also be argued that *good* MSP indicates that most of the deterministic dynamics of the system under study have been modeled. It is in this spirit, and by extrapolation of the aforementioned arguments to the nonlinear domain, that it is desired to design predictors capable of accurate MSP.

III.3.1 Use of the FMLP as a Nonlinear Predictor

If the interest is in utilizing the FMLP as a nonlinear predictor, then it can be configured so that its input and output layer represent the measured and predicted system inputs and outputs, respectively. If $u(k-i) \in R^r$ and $y(k-j) \in R^m$, for

$i = 1, 2, \dots, n_u$, $j = 1, 2, \dots, n_y$ are delayed inputs and outputs, respectively, then the inputs to the first FMLP layer, i.e. the inputs to the network, can be defined by the following $N(1)$ -dimensional vector:

$$\mathbf{x}_{[1]}(k) = [y(k-1), \dots, y(k-n_y), u(k-1), \dots, u(k-n_u)]^T, \quad (8)$$

where it is assumed that an equal number of delays has been used for each element of the input and output vectors. Considering the special structure of the input and output layers, and in view of equation (5), the equations for a single-hidden-layer FMLP network can be expressed as follows:

$$x_{[2,j]}(k) = F_{[2]} \left(\sum_{n=1}^{N(1)} w_{[1,n][2,j]} x_{[1,n]}(k) + b_{[2,j]} \right), \quad (9)$$

$$\hat{y}_i(k/k-1) \equiv x_{[3,i]}(k) = \sum_{j=1}^{N(2)} w_{[2,j][3,i]} x_{[2,j]}(k) + b_{[3,i]}, \quad (10)$$

for $j = 1, \dots, N(2)$ and $i = 1, \dots, N(3)$, where for an m -output system $N(3) = m$, and where $N(2)$ is the number of hidden-layer nodes. Equations (9) and (10) can now be combined in the following compact form:

$$\hat{y}_i(k/k-1) = f_i(y(k-1), \dots, y(k-n_y), u(k-1), \dots, u(k-n_u)), \quad (11)$$

for $i = 1, \dots, m$, which is in the form of a single-step-ahead nonlinear (or NARX) predictor [7],[16].

The same argument can be extended to a network with multiple hidden layers, and thus a general FMLP network can be configured in the form of a single-step-ahead NARX predictor depicted by equation (11). Therefore CNNs depicted by equations (9) and (10) can be used as nonlinear, input-output model structures for SI. Such a

predictor is depicted in Figure 7c. It has been widely reported that the success of the FMLP networks in the input-output identification of nonlinear systems is partially because such an approximation uses certain nonlinearities, e.g. hyperbolic tangent, sigmoid, etc., as basis functions. In fact, it is well-known that FMLPs are simply complex curve-fitting tools, allowing global approximation of almost all nonlinear functions that are of any practical use. Similar convictions about the functionality of FMLP networks as good curve-fitting tools have been expressed by numerous other researchers [37]. In fact, in a number of recent studies it has been rigorously proven that an FMLP network with one hidden layer is sufficient for approximating a large class of nonlinear functions. However, no information is provided regarding the number of regressors or nodes needed in the hidden layer [17].

In the absence of high quality sensor readings, then the predictor of equation (11) can also be utilized in the following (p-1)-step-ahead predictor form:

$$\begin{aligned} \hat{y}_i(k+p/k-1) = & f_i(\hat{y}(k+p-1/k-1), \dots, \hat{y}(k+p-n_y/k-1), \\ & u(k+p-1), \dots, u(k+p-n_u)). \end{aligned} \quad (12)$$

Identification of a model in the form of equation (11) is referred to as series-parallel identification in the SI literature, whereas identification of a model in the form of equation (12) is referred to as parallel identification [37]. Equation (12) can be used

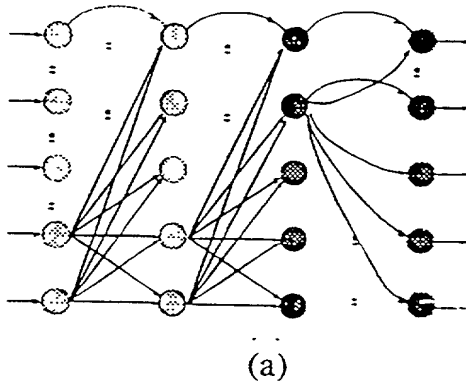
for performing MSP, recursively, while equation (11) is primarily used for SSP. The predictor of equation (12) is depicted in Figure 7e. It should be noted that equations (11) and (12) both imply availability of the inputs without any uncertainty. However, if there is some uncertainty associated with future values of the inputs, the $u(.)$'s in both equations (11) and (12) are replaced with $\hat{u}(.)$'s. Furthermore, as widely reported in the literature, even though availability of predictors in form of equation (12) is desirable for a number of practical reasons, the difficulties associated with designing accurate predictors in this form has limited their development and use [37].

The parameters of the FMLP model structure, either in form of equation (11) or (12), that is the weights and the biases can be iteratively estimated using the BP algorithm, or any of its variants, such as the Adaptive Back Propagation (ABP) algorithm [46].

III.3.2 Use of the RMLP as a Nonlinear Predictor

The RMLP was first proposed for use in nonlinear SI in 1990 [20]. Since then, it has been extensively studied, analyzed, and tested on numerous real-world applications. Furthermore, a learning algorithm for its effective training have been developed. An attractive feature of the RMLP is the explicit distinction among its layer, visually simplifying the separation of the feedforward and the feedback parts of the network. Although there is a substantial benefit gained by considering a locally recurrent network architecture for SI, as compared to using a purely feedforward one, there is, however, an increase in the complexity of the required learning. This is primarily attributed to the increasingly complex dynamic behavior of a recurrent network, the

FMLP



RMLP

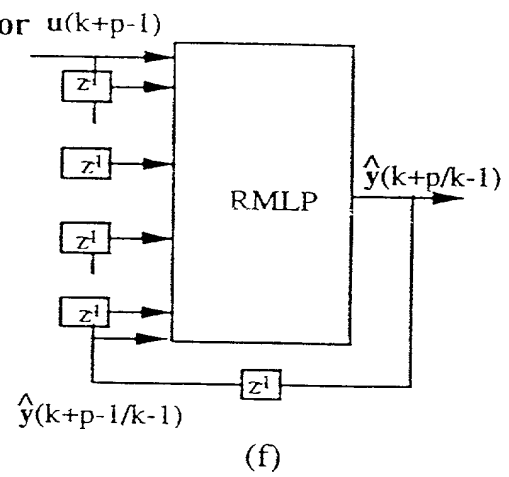
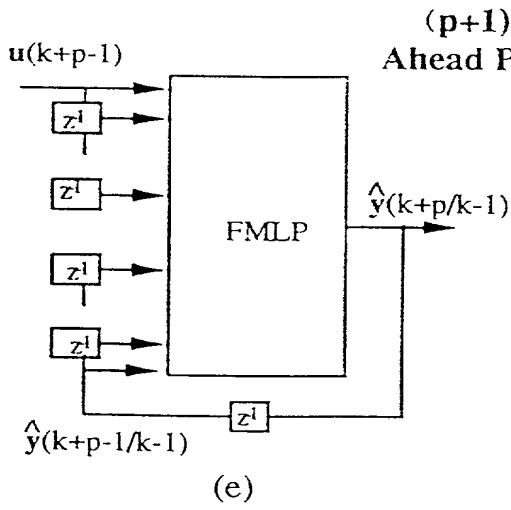
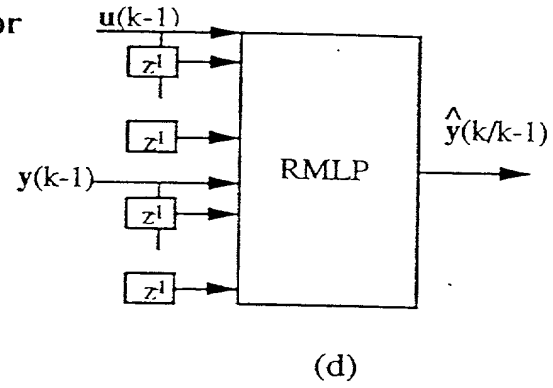
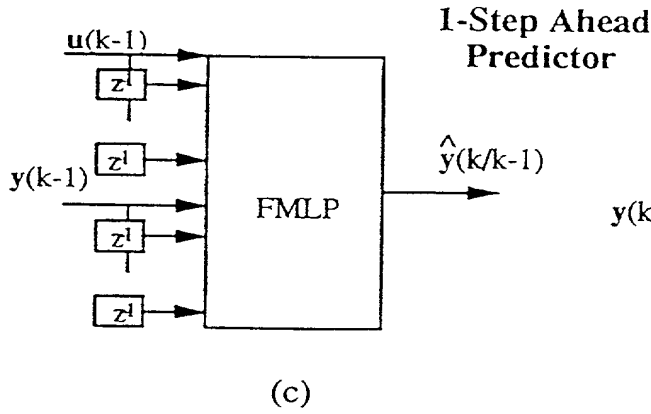
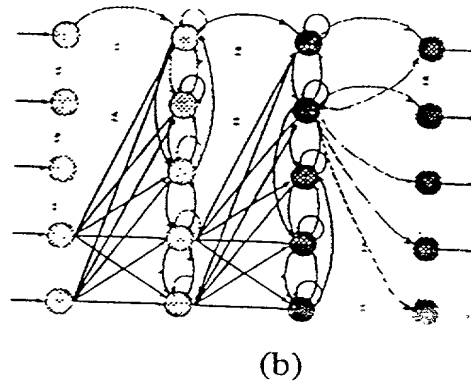


Figure 7 Summary of Predictor Architectures.

increased number of adjustable parameters for the same number of hidden nodes, and the issues associated with network stability during learning [16].

Another attractive feature of the RMLP, as well as any other dynamic recurrent neural network, is the fact that temporal effects can be inherently incorporated in it. As a result the RMLP can be useful for approximating dynamic nonlinearities, i.e. nonlinearities with memory effects, as opposed to static nonlinearities. Dynamic systems are characterized by nonlinearities with explicit temporal effects and they cannot be accurately approximated by purely feedforward networks such as FMLPs even if GF is used as previously discussed. With the appropriate choice of the number of layers and nodes, it is believed that an RMLP can approximate the dynamics of many complex system of interest in practical applications. Note, however, that there is no rigorous mathematical proof to justify such a claim. Experience, though, with numerous complex systems has partially established the validity of this claim.

As with the FMLP, the inputs to the first layer of an RMLP, i.e. the network inputs, are denoted by $x_{[1]}$, and the output of the last layer, i.e. the network outputs, are denoted by $x_{[L]}$. The input layer of an RMLP network acts as a buffer for the input stream, whereas the hidden (discriminatory) layers enable the break-down of the functional space to be identified into regions where it can be approximated by piecewise components. This is achieved by using a saturation function (usually sigmoid or a hyperbolic tangent).

If the RMLP is to be utilized as a nonlinear predictor, then its input and output layers must be appropriately defined. Because of the ever present uncertainties, predictors must be designed to operate in a closed-loop manner. This is the case with an

RMLP-based predictor. Let the input layer of the RMLP be defined by the following $N(1)$ -dimensional vector; as in the case of an FMLP (equation (8)):

$$\mathbf{x}_{[1]}(k) = [\mathbf{y}(k-1), \dots, \mathbf{y}(k-n_y), \mathbf{u}(k-1), \dots, \mathbf{u}(k-n_u)]^T, \quad (13)$$

where it is assumed that an equal number of delays has been used for each element of the input and output vectors. Furthermore, it is assumed that the process under consideration has r inputs and m outputs, i.e. $\mathbf{u}(k) \in R^r$ and $\mathbf{y}(k) \in R^m$.

Considering the special structure of the input and output layers, and in view of equations (6) and (7), the equations for a single-hidden-layer RMLP network can be expressed as follows:

$$x_{[1,j]}(k) = z_{[1,j]}(k); j = 1, \dots, N(1), \quad (14)$$

$$z_{[2,i]}(k) = \sum_{n=1}^{N(2)} w_{[2,n][2,i]} x_{[2,n]}(k-1) + \sum_{n=1}^{N(1)} w_{[1,n][2,i]} x_{[1,n]}(k) + b_{[2,i]}, \quad (15)$$

$$x_{[2,i]}(k) = F_{[2]}(z_{[2,i]}(k)); i = 1, \dots, N(2), \quad (16)$$

$$x_{[3,p]}(k) = \sum_{n=1}^{N(2)} w_{[2,n][3,p]} x_{[2,n]}(k) - b_{[3,p]}; p = 1, \dots, N(3). \quad (17)$$

where, $N(1) = m \times n_y + r \times n_u$ and $N(3) = m$. Furthermore, define

$$\hat{y}_p(k/k-1) \equiv x_{[3,p]}(k); p = 1, \dots, m.$$

Now, eliminating $x_{[1,j]}(k)$, $x_{[2,i]}(k)$, and $x_{[3,p]}(k)$ from equation (14) through (17), the following forms are obtained:

$$\begin{aligned} z_{[2,j]}(k) = & \sum_{n=1}^{N(2)} w_{[2,n][2,j]} F_{[2]}(z_{[2,n]}(k-1)) - \sum_{n=1}^{N(1)} w_{[1,n][2,j]} z_{[1,n]}(k) \\ & + b_{[2,j]}; j = 1, \dots, N(2), \end{aligned} \quad (18)$$

$$\hat{y}_p(k/k-1) = \sum_{n=1}^{N(2)} w_{[2,n][3,p]} F_{[2]}(z_{[2,n]}(k)) - b_{[3,p]}, p = 1, \dots, N(3). \quad (19)$$

Now, defining:

$$\hat{\mathbf{y}}(k/k-1) = [\hat{y}_1(k/k-1), \dots, \hat{y}_{N(3)}(k/k-1)]^T, \quad (20)$$

$$\mathbf{z}_{[1]}(k) = \mathbf{x}_{[1]}(k) \equiv \mathbf{U}(k-1), \quad (21)$$

$$\mathbf{z}_{[2]}(k) = [z_{[2,1]}(k), \dots, z_{[2,N(2)]}(k)]^T. \quad (22)$$

$$\mathbf{b}_{[2]} = [b_{[2,1]}, \dots, b_{[2,N(2)]}]^T. \quad (23)$$

$$\mathbf{b}_{[3]} = [b_{[3,1]}, \dots, b_{[3,N(3)]}]^T, \quad (23)$$

$$\mathcal{W}_{1 \rightarrow 2} = [w_{[1,i][2,j]}; i = 1, \dots, N(1); j = 1, \dots, N(2)], \quad (25)$$

$$\mathcal{W}_{2 \rightarrow 2} = [w_{[2,i][2,j]}; i = 1, j = 1, \dots, N(2)], \quad (26)$$

$$\mathcal{W}_{2 \rightarrow 3} = [w_{[2,i][3,j]}; i = 1, \dots, N(2); j = 1, \dots, N(3)], \quad (27)$$

equations (18) and (19) can be written in the following compact form :

$$\mathbf{z}_{[2]}(k) = \mathcal{W}_{2 \rightarrow 2} \mathbf{F}_{[2]}(\mathbf{z}_{[2]}(k-1)) + \mathcal{W}_{1 \rightarrow 2} \mathbf{U}(k-1) + \mathbf{b}_{[2]}, \quad (28)$$

$$\hat{\mathbf{y}}(k/k-1) = \mathcal{W}_{2 \rightarrow 3} \mathbf{F}_{[2]}(\mathbf{z}_{[2]}(k)) - \mathbf{b}_{[3]}, \quad (29)$$

where $\mathbf{F}_{[2]}(\cdot)$ is the following vector function:

$$\mathbf{F}_{[2]}(\cdot) = [F_{[2]}^1(\cdot), \dots, F_{[2]}^{N(2)}(\cdot)]^T. \quad (30)$$

Equations (28) and (29) are of the form:

$$\begin{cases} \mathbf{z}_{[2]}(k) = \mathcal{F}(\mathbf{z}_{[2]}(k-1)) + \mathcal{W}_{1 \rightarrow 2} \mathbf{U}(k-1), \\ \hat{\mathbf{y}}(k/k-1) = \mathcal{H}(\mathbf{z}_{[2]}(k)), \end{cases} \quad (31)$$

which is in the form of a nonlinear (empirical) state-space predictor, as reported in Ljung [34]. It is an empirical state-space because the state vector $\mathbf{z}_{[2]}(k)$ has no particular physical interpretation. This predictor is depicted in Figure 7d. Similarly to equation (11) of page 44, the predictor of equation (31) can be utilized for SSP.

By replacing $\mathbf{U}(k-1)$ with $\hat{\mathbf{U}}(k+p-1)$ defined as follows:

$$\begin{aligned} \hat{\mathbf{U}}(k+p-1) \equiv & [\hat{\mathbf{y}}(k+p-1/k-1), \dots, \hat{\mathbf{y}}(k-p-n_y/k-1), \\ & \dots, \mathbf{u}(k+p-1), \dots, \mathbf{u}(k+p-n_u)]^T, \end{aligned} \quad (32)$$

the state-space predictor of equation (31) can be utilized in the following $(p+1)$ -step predictor form:

$$\begin{cases} \mathbf{z}_{[2]}(k+p) = \mathcal{F}(\mathbf{z}_{[2]}(k+p-1)) + \mathcal{W}_{1 \rightarrow 2} \hat{\mathbf{U}}(k+p-1), \\ \hat{\mathbf{y}}(k-p/k-1) = \mathcal{H}(\mathbf{z}_{[2]}(k+p)). \end{cases} \quad (33)$$

The arguments made regarding the input-output parallel identification model (equation (12)) are also applicable for the case of the predictor given by equation (33). A block diagram for this form of the predictor is given in Figure 7f. We now turn our intention to the learning (parameter estimation) algorithm for the predictors given by equations (31) and (33). Learning algorithms for the predictors given by equations (11) and (12) are available in the literature, and they are not repeated in this document [37],[56].

III.4 Recurrent Multilayer Perceptron Learning Algorithm with Teacher Forcing

In both static and dynamic SI, whether using CNNs or not, the objective is to determine a preferably adaptive algorithm, or rule, which adjusts the parameters of

the network, based on the information contained in a given set of system observations. When dealing with perceptron-type CNNs, BP learning is the most commonly used learning method. It was first derived for static networks by Werbos [65], and later it was extended to incorporate temporal effects under the name of Backpropagation-Through-Time (BTT). Documentation on standard BP can be found in many books and papers which discuss CNNs. There have been numerous variations to the standard BP algorithm reported in the literature, however, no attempt has been made in this study to document any of these approaches or to use them. For a brief review the reader is referred to Parlos et. al [46].

The learning algorithm for RMLP with TF is a gradient descent algorithm minimizing a mean-squared-error (MSE) objective function. TF means that the inputs to the network include up to the latest sensed inputs(s) and output(s) of the system. In other words, at each time-step the network is updated with the latest observation(s). The basic mechanism of this learning rule is the adjustment of the network weights and the bias terms, until the MSE between the output predicted by the network and the sensed system output target is less than a prespecified tolerance.

The training set can be expressed as follows:

$$S = \{(u_i(k), y_j(k)), \quad \forall k = 1, \dots, NP; i = 1, \dots, N(1); j = 1, \dots, N(L)\}. \quad (34)$$

where NP is the total number of data pairs in the training set. Specifically, the objective of most supervised learning algorithms is to determine the change in the

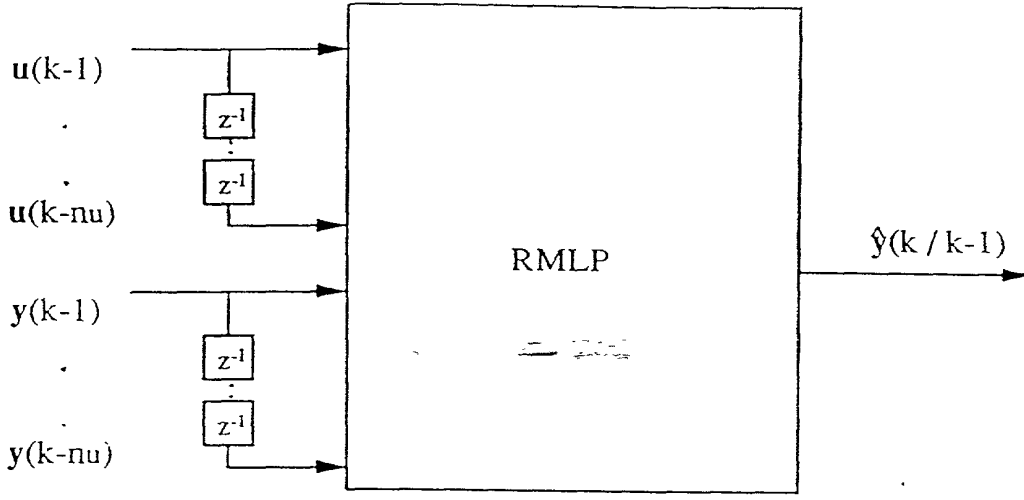


Figure 8. An RMLP Utilized as a Teacher Forcing Predictor.

network parameters $w_{[l-1,i][l,j]}$, $w_{[l,i][l,j]}$ and $b_{[l,i]}$, for all i, j and l , such that some form of the following functional:

$$E \equiv \frac{1}{2} \sum_{k=1}^{NP} E(k) \equiv \frac{1}{2} \sum_{k=1}^{NP} \sum_{j=1}^{N(L)} (\hat{y}_j(k/k-1) - y_j(k))^2, \quad (35)$$

is minimized. Figure 8 is a representation of an RMLP configured for use as a predictor with TF. In this figure $\hat{y}(k/k+1)$ is the predicted output using the RMLP, while $y(k-1)$ is the latest observed (sensed) system output.

Two learning modes can be utilized to train an RMLP. The off-line learning mode, and the on-line learning mode. For the former, assuming that the training set contains

NP pairs of input-output data which have been selected before initiating the training, these pairs are repetitively presented to the network until it reproduces them to within a desired error tolerance. During such training sessions, the network weights and biases are updated using the following gradient descent rules:

$$\Delta w_{[l-1,j][l,i]} = -\eta \sum_{k=1}^K \left(\frac{\partial E(k)}{\partial w_{[l-1,j][l,i]}} \right), \quad (36)$$

$$\Delta w_{[l,j][l,i]} = -\eta \sum_{k=1}^K \left(\frac{\partial E(k)}{\partial w_{[l,j][l,i]}} \right), \quad (37)$$

$$\Delta b_{[l,i]} = -\eta \sum_{k=1}^K \left(\frac{\partial E(k)}{\partial b_{[l,i]}} \right), \quad (38)$$

where K can be set to 1 or NP for individual or batch update, respectively, and where η is the learning rate, interactively set and altered by the user for proper convergence. For the on-line learning mode, however, there is no predetermined training set, and the weight updating must be performed as sensed information is received.

For the dynamic gradient descent learning, the prediction error is defined by equation (35). The error gradient with respect to the weights and the biases can be obtained by using the chain-rule, as follows:

$$\frac{\partial E(k)}{\partial w_{[l,j][l,i]}} = 2 \sum_{n=1}^{N(\mathcal{L})} (\hat{y}_n(k/k-1) - y_n(k)) \frac{\partial \hat{y}_n(k/k-1)}{\partial w_{[l,j][l,i]}}, \quad (39)$$

$$\frac{\partial E(k)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{n=1}^{N(\mathcal{L})} (\hat{y}_n(k/k-1) - y_n(k)) \frac{\partial \hat{y}_n(k/k-1)}{\partial w_{[l-1,j][l,i]}}, \quad (40)$$

$$\frac{\partial E(k)}{\partial b_{[l,i]}} = 2 \sum_{n=1}^{N(\mathcal{L})} (\hat{y}_n(k/k-1) - y_n(k)) \frac{\partial \hat{y}_n(k/k-1)}{\partial b_{[l,i]}}. \quad (41)$$

As an example for an updated process of the feedforward weights, consider the gradient term $\frac{\partial \hat{y}_n(k/k-1)}{\partial w_{[l,j][l,i]}}$. Differentiating equation (6) and equation (7) with respect to

$w_{[l,j][l,i]}$, it can be seen that $\frac{\partial \hat{y}_n(k/k-1)}{\partial w_{[l,j][l,i]}}$ can be obtained in terms of $\frac{\partial \hat{y}_m(k-1/k-2)}{\partial w_{[l,j][l,i]}}$ and in terms of $\frac{\partial x_{[l-1,m]}(k/k-1)}{\partial w_{[l,j][l,i]}}$ for all m . Similarly, to obtain $\frac{\partial x_{[l-1,m]}(k)}{\partial w_{[l,j][l,i]}}$ first $\frac{\partial x_{[l-1,p]}(k-1)}{\partial w_{[l,j][l,i]}}$ and $\frac{\partial x_{[l-2,p]}(k)}{\partial w_{[l,j][l,i]}}$, for all p , must be evaluated. Therefore, the approach considered is to first evaluate $\frac{\partial x_{[2,m]}(k)}{\partial w_{[l,j][l,i]}}$, the output gradients of the second layer. Then, the algorithm propagates forward, in the process evaluating the output gradients of the subsequent layers, until $\frac{\partial \hat{y}_n(k/k-1)}{\partial w_{[l,j][l,i]}}$ is obtained. At that point the error gradients can be evaluated using equation (39) [16],[42].

The recursion equations used to execute the forward gradient propagation can be derived by differentiating equation (6) and (7) with respect to $w_{[l,j][l,i]}$, $w_{[l-1,j][l,i]}$ and $b_{[l,i]}$, respectively. These differentiations result in the following equations:

$$\frac{\partial x_{[l',n]}(k)}{\partial w_{[l,j][l,i]}} = \begin{cases} F'_{[l']}(z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l,j][l,i]}} \right] & \text{if } l' > l, \\ F'_{[l']}(z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[l',j]}(k-1) \right] & \text{if } l' = l, \\ 0 & \text{if } l' < l, \end{cases} \quad (42)$$

$$\frac{\partial x_{[l',n]}(k)}{\partial w_{[l-1,j][l,i]}} = \begin{cases} F'_{[l']}(z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } l' > l, \\ F'_{[l']}(z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[l'-1,j]}(k) \right] & \text{if } l' = l, \\ 0 & \text{if } l' < l, \end{cases} \quad (43)$$

$$\frac{\partial x_{[l',n]}(k)}{\partial b_{[l,i]}} = \begin{cases} F'_{[l']}(z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial b_{[l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l,j][l,i]}} \right] & \text{if } l' > l, \\ F'_{[l']}(z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial b_{[l,i]}} \right. \\ \left. + \delta_{in} \right] & \text{if } l' = l, \\ 0 & \text{if } l' < l, \end{cases} \quad (44)$$

for all layers $l', l = 1, \dots, \mathcal{L}$, and $i, j = 1, \dots, N(l)$, sweeping the entire network, where

$$\delta_{in} = \begin{cases} 1 & \text{if } i = n, \\ 0 & \text{otherwise,} \end{cases} \quad (45)$$

and where the initial values for the gradients $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l-1,j][l,i]}}$ and $\frac{\partial x_{[l',n]}(0)}{\partial b_{[l,i]}}$ are set to zero. Equations (42) through equation (44) must be applied separately for each weight, and bias in the network. However, it is sufficient to always start from layer $l' = l$, and then propagate forwards until $l' = \mathcal{L}$, because $\frac{\partial x_{[l',n]}(k)}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}(k)}{\partial w_{[l-1,j][l,i]}}$ and $\frac{\partial x_{[l',n]}(k)}{\partial b_{[l,i]}}$ equal 0 for $l' < l$ [16],[42].

III.5 Recurrent Multilayer Perceptron Learning Algorithm with Global Feedback

The learning algorithm with GF is also a gradient descent algorithm minimizing a MSE objective function. GF means that the inputs utilized by the network include up to and including the latest predicted output(s) generated by the CNN. Figure 9 is a representation of the RMLP with GF, also depicted by equation (33) for a single hidden layer. In this figure $\hat{y}(k + p/k - 1)$ is the output predicted using the CNN, while $\hat{y}(k + p - 1/k - 1)$ is the output predicted by the CNN at the previous time-step.

The fact that we have past predicted CNN outputs utilized in GF, as compared to having past sensed system output in TF, makes the gradients derivation in GF more

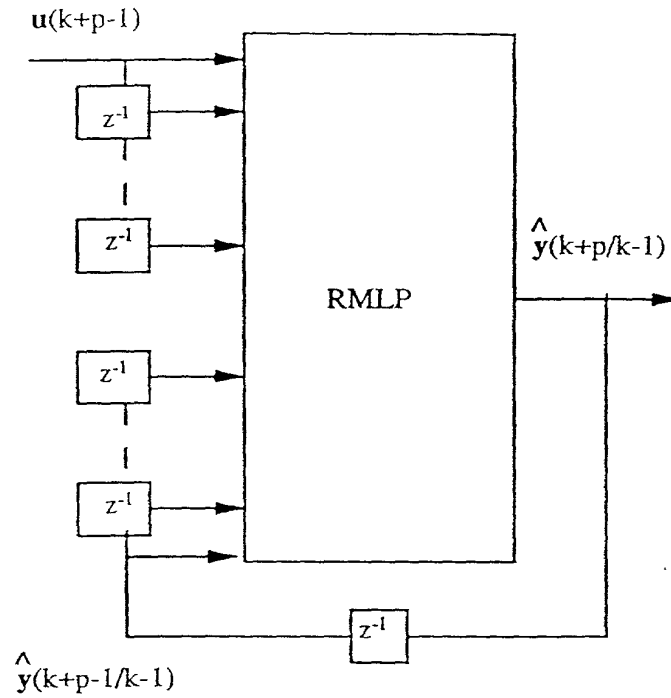


Figure 9. An RMLP with Global Feedback Utilized as a $(p+1)$ -Step-Ahead Predictor.

complex than in TF. When using GF, the error gradients each time-step, with respect to some of the weights and biases at depend on the values of the same gradients at the previous time-step. Thus, by recursively computing the error gradients, their dependence to all previous values of the error gradients is implicitly ensured. This feature enables the CNN to perform accurate MSP on the training set, as well as any other set not seen by the network, depending on the procedure followed in the predictor design process. This is because the objective function being minimized consists of the MSP error of the training or testing set. Specifically, consider the

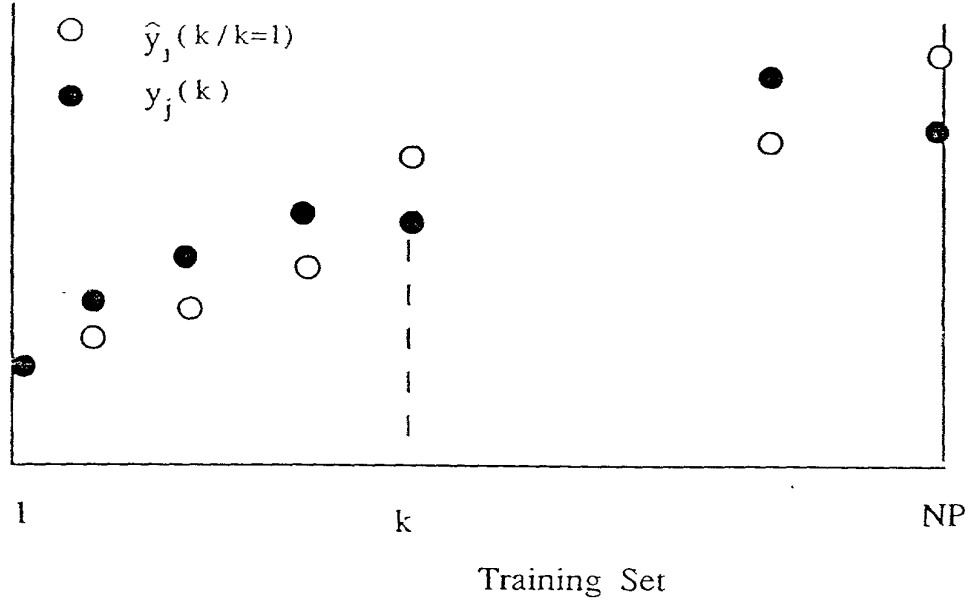


Figure 10. MSP within The Training Set.

following error functional based on equation (35).

$$E \equiv \frac{1}{2} \sum_{k=1}^{NP} E(k) \equiv \frac{1}{2} \sum_{k=1}^{NP} \sum_{j=1}^{N(L)} (\hat{y}_j(k/k=1) - y_j(k))^2. \quad (46)$$

where, $\hat{y}_j(k/k=1)$ is the recursively predicted CNN output, for $k = 1, \dots, NP$, and $y_j(k)$ is the k^{th} target (sensed output) in the training set (see Figure 10). In view of equation (33) and the representation of Figure 9, it is clear that every predicted output $\hat{y}_j(k/k=1)$ in the training set indirectly depends on all previous predictions $\hat{y}_j(k-1/k=1), \hat{y}_j(k-2/k=1), \dots, \hat{y}_j(k-n_y/k=1)$. Considering a single hidden layer for simplicity, equation (33) can be utilized to express this dependence as follows:

$$\begin{cases} \hat{y}(2/k = 1) = \mathcal{H}(z_{[2]}(2)), \\ z_{[2]}(2) = \mathcal{F}(z_{[2]}(1)) + \mathcal{W}_{1 \rightarrow 2} \hat{U}(1), \\ \hat{U}(1) = [y(1), 0, \dots, u(1), 0, \dots, 0]^T, \end{cases} \quad (47)$$

$$\begin{cases} \hat{y}(k - 1/k = 1) = \mathcal{H}(z_{[2]}(k - 1)), \\ z_{[2]}(k - 1) = \mathcal{F}(z_{[2]}(k - 2)) + \mathcal{W}_{1 \rightarrow 2} \hat{U}(k - 2), \\ \hat{U}(k - 2) = [\hat{y}(k - 2/k = 1), \dots, \hat{y}(k - n_y - 1/k = 1), \\ \quad u(k - 2), \dots, u(k - n_u - 1)]^T, \end{cases} \quad (48)$$

and

$$\begin{cases} \hat{y}(k/k = 1) = \mathcal{H}(z_{[2]}(k)), \\ z_{[2]}(k) = \mathcal{F}(z_{[2]}(k - 1)) + \mathcal{W}_{1 \rightarrow 2} \hat{U}(k - 1), \\ \hat{U}(k - 1) = [\hat{y}(k - 1/k = 1), \dots, \hat{y}(k - n_y/k = 1), \\ \quad u(k - 1), \dots, u(k - n_u)]^T. \end{cases} \quad (49)$$

Note that the zero entries in the above equations can be replaced with the values of the target at $k = 1$, should such values represent steady-state conditions.

Similar concepts are used in BTT by Werbos [56],[64]. In BTT, the gradients at time-step $k + 1$ depend on all the gradient from time-step k back to $k = 1$. For example, if there are 100 data points in a training set, then the gradients computed at the 87th data point are based on all the gradients from the 86th data point all the way to the gradients of the first data point. In a sense the BP is carried through time. The gradients derivation in BTT is less complex than in the RMLP with GF, though the algorithm is computationally more intensive because for each past time-step dependence of the gradients, an additional CNN layer is added. For example, in the aforementioned example there would be as many as 100 layers involved in the

gradients computations. It has been our experience that the RMLP with GF approach has been sufficient for a CNN to learn the dynamics of complex process systems encountered in practice, and such a predictor should have good MSP performance.

In order to derive the error gradients for the RMLP with GF, without having to unfold the network as it is done in BTT, it is necessary to introduce an additional assumption. If the predictor set-up of Figure 9 is applied to the training set depicted by Figure 10, then it is observed that a number of CNN inputs are correlated with each other. For example, $\hat{y}(k-1/k=1)$ is dependent upon $\hat{y}(k-2/k=1)$ and so on. This complication can be circumvented in two ways; either by assuming that $\hat{y}(k-1/k=1), \dots, \hat{y}(k-n_y+1/k=1)$ are outputs of other RMLPs, i.e. unfolding the network, or by assuming that

$$\begin{cases} \hat{y}(k-2/k=1) = y(k-2), \\ \vdots \\ \hat{y}(k-n_y/k=1) = y(k-n_y). \end{cases} \quad (50)$$

That is all past predictions, with the exception of the latest prediction $\hat{y}(k-1/k=1)$, are replaced by their equivalent targets from the training set. Thus, the RMLP is composed of two types of inputs nodes: those that consist of all independent inputs, i.e. $y(k-2), \dots, y(k-n_y), u(k-1), \dots, u(k-n_u)$, and the prediction node $\hat{y}(k-1/k=1)$. Note that even with the aforementioned assumption, it is still the MSP error that is computed for the training (and testing) set.

In the following development we make the assumption that equation (50) is valid. Then, the equations describing the i th node of the first hidden layer of the RMLP network using GF, for the k^{th} sample of the training set defined by equation (34), are given by the following difference equations:

$$\begin{aligned}
z_{[2,i]}(k) &= \sum_{j=1}^{N(2)} w_{[2,j][2,i]} x_{[2,j]}(k-1) + \sum_{j=1}^{N(1)-N(\mathcal{L})} w_{[1,j][2,i]} x_{[1,j]}(k) \\
&+ \sum_{j=N(1)-N(\mathcal{L})+1}^{N(1)} w_{[1,j][2,i]} \hat{y}_j(k-1/k=1) + b_{[2,i]}, \tag{51}
\end{aligned}$$

and,

$$x_{[2,i]}(k) = F_{[2]}(z_{[2,i]}(k)). \tag{52}$$

The equations describing the i th node of the l th layer, other than the first hidden layer, of the RMLP network using GF are given by the following difference equation:

$$z_{[l,i]}(k) = \sum_{j=1}^{N(l)} w_{[l,j][l,i]} x_{[l,j]}(k-1) + \sum_{j=1}^{N(l-1)} w_{[l-1,j][l,i]} x_{[l-1,j]}(k) + b_{[l,i]}, \tag{53}$$

and

$$x_{[l,i]}(k) = F_{[l]}(z_{[l,i]}(k)), \tag{54}$$

for $l = 2, \dots, N(\mathcal{L})$. Note that comparing equation (51) with equation (53), an additional third term is present which accounts for the latest CNN prediction utilized in the input layer.

For the dynamic gradient descent learning with GF, the prediction error is defined by equation (46). The error gradient with respect to the weights and the biases can be obtained by using the chain-rule, as follows:

$$\frac{\partial E(k)}{\partial w_{[l,j][l,i]}} = 2 \sum_{n=1}^{N(\mathcal{L})} (\hat{y}_n(k/k=1) - y_n(k)) \frac{\partial \hat{y}_n(k/k=1)}{\partial w_{[l,j][l,i]}}, \tag{55}$$

$$\frac{\partial E(k)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{n=1}^{N(\mathcal{L})} (\hat{y}_n(k/k=1) - y_n(k)) \frac{\partial \hat{y}_n(k/k=1)}{\partial w_{[l-1,j][l,i]}}, \tag{56}$$

$$\frac{\partial E(k)}{\partial b_{[l,i]}} = 2 \sum_{n=1}^{N(\mathcal{L})} (\hat{y}_n(k/k=1) - y_n(k)) \frac{\partial \hat{y}_n(k/k=1)}{\partial b_{[l,i]}}. \tag{57}$$

Note that the last three equations are similar to the ones given for the case of TF learning algorithm. The recursion equations used to execute the forward gradient propagation can be derived by differentiating equations (51) through (54) with respect to $w_{[l,j][l,i]}$, $w_{[l-1,j][l,i]}$, and $b_{[l,i]}$, respectively. These differentiations result in equations described in the following development [1]. For the gradients with respect to the network connections within the layer, namely, the recurrent and cross-talk weights, we have the following equation, valid for all hidden layers and output layer, namely, $1 < l' \leq L$:

$$\frac{\partial x_{[l',n]}(k)}{\partial w_{[l,j][l,i]}} = \begin{cases} F'_{[l']} (z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l,j][l,i]}} \right] & \text{if } l' \neq l, \\ F'_{[l']} (z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l,j][l,i]}} - \delta_{in} x_{[l',j]}(k-1) \right] & \text{if } l' = l, \end{cases} \quad (58)$$

For interlayer connections, namely $\frac{\partial x_{[l',n]}(k)}{\partial w_{[l-1,j][l,i]}}$, we have the following gradient which is computed somewhat differently than the previous gradient. For the first hidden layer ($l' = 2$), the gradient is given by the following equations:

$$\frac{\partial x_{[2,n]}(k)}{\partial w_{[l-1,j][l,i]}} = \begin{cases} F'_{[2]} (z_{[2,n]}(k)) \left[\sum_{m=1}^{N(2)} w_{[2,m][2,n]} \frac{\partial x_{[2,m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(1)-N(L)} w_{[1,m][2,n]} \frac{\partial x_{[1,m]}(k)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } 1 \leq j \leq L, l \neq 2, \\ F'_{[2]} (z_{[2,n]}(k)) \left[\sum_{m=1}^{N(2)} w_{[2,m][2,n]} \frac{\partial x_{[2,m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(1)-N(L)} w_{[1,m][2,n]} \frac{\partial x_{[1,m]}(k)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[1,j]}(k) \right] & \text{if } 1 \leq j \leq L, l = 2, \\ F'_{[2]} (z_{[2,n]}(k)) \left[\sum_{m=1}^{N(2)} w_{[2,m][2,n]} \frac{\partial x_{[2,m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=N(1)-N(L)+1}^{N(1)} w_{[1,m][2,n]} \frac{\partial y_m(k-1)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } L < j \leq N(1), \forall l, \end{cases} \quad (59)$$

where $L \equiv N(1) - N(\mathcal{L})$.

For all hidden layers other than the first hidden layer, namely $2 < l' \leq \mathcal{L}$, the gradient with respect to interlayer connections is computed as follows:

$$\frac{\partial x_{[l',n]}(k)}{\partial w_{[l-1,j][l,i]}} = \begin{cases} F'_{[l']} (z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } l' \neq l, \\ F'_{[l']} (z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l-1,j][l,i]}} \right] \\ + \delta_{in} x_{[l'-1,j]}(k) & \text{if } l' = l. \end{cases} \quad (60)$$

The gradients with respect to the biases are given by the following equation:

$$\frac{\partial x_{[l',n]}(k)}{\partial b_{[l,i]}} = \begin{cases} F'_{[l']} (z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial b_{[l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l,j][l,i]}} \right] & \text{if } l' > l, \\ F'_{[l']} (z_{[l',n]}(k)) \left[\sum_{m=1}^{N(l')} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(k-1)}{\partial b_{[l,i]}} \right. \\ \left. + \sum_{m=1}^{N(l'-1)} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(k)}{\partial w_{[l,j][l,i]}} + \delta_{in} \right] & \text{if } l' = l. \end{cases} \quad (61)$$

Equations (58) through (61), are valid for all layers l' , $l = 2, \dots, \mathcal{L}$, and nodes $i, j = 1, \dots, N(l)$, sweeping the entire network, and where,

$$\delta_{in} = \begin{cases} 1 & \text{if } i = n, \\ 0 & \text{otherwise.} \end{cases} \quad (62)$$

Note that $N(1)$ includes the number of current and delayed inputs of the system as well as the number of delayed output(s) and prediction utilized as input to the network. Furthermore, the initial values for the gradients $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l-1,j][l,i]}}$ and $\frac{\partial x_{[l',n]}(0)}{\partial b_{[l,i]}}$ are set to zero. Equations (58) through equation (61) must be applied separately for each weight and bias in the network [1]. It should be also noted that

if the RMLP with GF is trained on-line, using a moving window of length $(p+1)$ as depicted in Figure 6, instead of a fixed training set of length NP , then the conditional predictions of the form $\hat{y}(. / k = 1)$ used throughout this section must be replaced by conditional predictions of the form $\hat{y}(. / k - 1)$.

III.6 Network Architecture Design

The ultimate objective of a model-based on CNNs, is to perform accurate MSP. MSP should be successful not only on the data used for training the CNN, but also on any other new data presented to the CNN. This is also known as the ability of the CNN to generalize (*generalization*) or extrapolate. A CNN model which performs accurate MSP for any input signals is a model which has learned (replicated or captured) most of the important deterministic dynamics of the system under study. In the linear SI literature, this issue has not been widely addressed. Therefore this challenge lies with a complex process system, like the U-Tube Steam Generator (UTSG) or other systems of similar complexity. Deriving appropriate learning algorithms is a necessary but not a sufficient condition. It is also necessary to design a predictive network of appropriate complexity. Network architecture design is an unresolved issue in the CNN community. Throughout this study, there is no claim made that this issue is resolved either. On the other hand, what is proposed in this study are some known but not so well publicized guidelines, as well as some new guidelines. These guidelines were successful for the purpose of this research study as will be seen throughout the rest of this dissertation.

The major problem is that the neural network risks overfitting the training data because it has too many degrees of freedom. In overfitting, the network's output fits the training data too closely. It models the noise, in addition to the underlying function we want it to approximate. The form of the mapping used by CNNs is so flexible and the gradient-descent learning process is so relentless that, unless it is prevented, overfitting is a serious hazard. Less powerful modeling techniques, such as linear regression, are also capable of overfitting. ~~But since~~ linear regression can only fit a straight line to the data and it cannot contort itself into complex forms that the neural network can, overfitting is less of a concern. So, by attempting CNN-based models one can do a better job than with a linear regression, but this strength is also a weakness because it can also lead to overfitting.

The solution to overfitting is therefore to limit the network's approximation capability. This can be done by allowing the network to provide a good fit on the training set, but without risking overfitting. The network approximation capability can be limited by controlling four variables: the number of hidden layers, the number of hidden nodes, the number of delayed inputs and/or past outputs, and the number of iterations prior to stopping the learning. Each of these main points will be addressed sequentially in the next few subsections.

III.6.1 Assignment of Networks Layers

Theoretically speaking, a single-hidden-layer FMLP is sufficient for modeling any linear or nonlinear system. The nonlinearity is provided by the hidden layer, through the sigmoid function. This is true for any Single Input Single Output (SISO) or Multi

Input Multi Output (MIMO) system. However, in the MIMO case, having a single hidden layer may not be sufficient, as it will be seen from the following analysis.

When it comes to modeling a dynamic system using a neural network, the degrees of freedom, N , of the network are of great importance. For each modeled system, there is an optimum number of the degrees of freedom for which a neural network model will learn all or most of its important dynamics. By degree of freedom, N , we mean the total number of weights and biases in the network. Assume, for example, that we have a system which has five inputs and one output. Also assume that the *optimum* degrees of freedom for modeling this system using a neural network is about 55, i.e. $N = 55$.

If we choose to have one hidden layer (in addition to the input layer and the output layer), then the neural network to choose from will be either 5-3-1 (with $N = 31$), or 5-4-1 ($N = 45$), or 5-5-1 ($N = 61$), or 5-6-1 ($N = 79$). As it can be seen from the latter architectures, the only suitable choice is the 5-5-1 network, because it has degree of freedom closest to the desired number of 55. If two hidden layers are included, then the networks to choose from will be either 5-3-2-1 (with $N = 42$), or 5-3-3-1 ($N = 52$), or 5-4-2-1 ($N = 57$), or 5-4-3-1 ($N = 68$). As it can be seen, from these architectures, there are two suitable choices: the 5-3-3-1 and the 5-4-2-1 networks. Both networks have degrees of freedom close to the desired value 55.

This simple example clearly shows that in some MIMO systems, setting the number of hidden layers to one may not yield the most effective neural network model. Note that the inclusion of additional hidden layers becomes more important, as the number

of system input(s) and/or or output(s) increases, including the delayed input(s) and output(s) considered as network inputs.

III.6.2 Assignment of Networks Nodes

Each weight and bias in the network is a degree of freedom that adds to the capacity of the network. The number of weights and biases determines the degrees of freedom that the network uses to fit the data. To limit the capacity of the network, we limit the number of weights. The number of weights in the network, however, is function of the number of nodes. As stated earlier, the objective of neural network modeling, is to design a predictor which does not overfit but it has good generalization performance. This can be done by choosing the number of nodes appropriately.

The rule of thumb in choosing the number of hidden nodes is to start with one or two hidden nodes and increase their number until an optimum number of nodes is reached. As it will be shown later in this research, only four hidden nodes and a single hidden layer were required to model a complex process system such as the U-Tube Steam Generator (UTSG) within a limited region of its operation. The effectiveness of the utilized learning algorithm has, of course, some impact on the resulting network architecture.

III.6.3 Assignment of Delayed Network Inputs and Outputs

The issue of having delayed inputs and delayed predicted or observed outputs utilized in the network input layer is problem dependent. In systems with reverse dynamic, for example, utilizing past inputs with one or two delays, becomes necessary for enabling the network to learn the dynamics of the system under study.

One approach for choosing the number of delayed inputs and outputs is to use the NARX structure detection algorithm [4]. This method attempts to detect all of the relevant terms to be used in the approximation. However, for complex systems with high process and sensor noise, the NARX structure detection algorithm may give unsatisfactory results. Another direct approach is to make use of any apriori physical knowledge about the system in determining the relative importance of delayed inputs, as well as delayed outputs.

III.6.4 Stopping Criterion

Stopping criterion is the method used to stop presentation of the training data to the network. It is another way by which the network approximation capability can be controlled. As the network is being trained, it passes from relatively simple to relatively complex mappings. This increase in complexity is not steady during training, but occurs in spurts. Each spurt adds a wrinkle or feature to the mapping function. Thus, the network passes through successive stages in which the number of hidden nodes that are really having some positive effect goes up one by one. As the network is further trained, and its mapping function grows more complex, it passes at some point through one configuration that gives the best generalization; after that, what the network learns amounts to overfitting. Training should be stopped at this crucial point after which any further training leads to overfitting.

Training data error cannot be used to determine the stopping criterion, because this error decreases always as a result of gradient descent learning. At some point in

the training, one of the hidden nodes finds a feature that is present in the training data but not true of the population in general; this is when the overfitting begins.

However, overfitting can be detected by observing the error on *testing* data. The procedure consist of having two independent sets of data: a training data set and a testing data set. Both data set should be collected from the same system under study. The network should be trained on the training set, but whenever the weights are updated the testing error should be computed. In general, the testing set error will be higher than the training set error. That is because the network is laboring to reduce the latter, and not the former. Both errors will initially decrease, but when the testing set error start increasing, overfitting begins. Consequently, the training should be stopped, and the weights which produced the lowest error on the testing data set should be selected. This process is called cross-validation (or out-of sample testing), and it is widely studied in the statistics literature.

III.7 Chapter Summary

The FMLP and the RMLP CNN architectures were both discussed in details in this chapter. The use of the FMLP and RMLP networks as nonlinear predictors was presented. Furthermore, the RMLP learning algorithms with TF and with GF were derived. Finally, some CNN architectural design issues were briefly discussed.

CHAPTER IV

A COMPARISON OF CONVENTIONAL AND COMPUTATIONAL NEURAL NETWORK APPROACHES IN NONLINEAR SYSTEM IDENTIFICATION

IV.1 Introduction

In this chapter a case-study is presented to compare the conventional methods and the Computational Neural Network (CNNs) methods in nonlinear System Identification (SI). Few studies have been reported on the use of nonlinear SI approaches for improving the relative accuracy of conventional model structures, hence such approaches have not yet gained wide acceptance. There are some well-known models in the linear SI literature [34], such as ARX, ARMAX, Box-Jenkins and others, and as a result will not be discussed in this dissertation. However, the polynomial Nonlinear AutoRegressive with eXogenous Input (NARX) model structure will be discussed in more detail. The polynomial NARX is considered a powerful technique in nonlinear SI, and it has proven successful in many applications, with the exception of highly complex process system with high process and sensor noise. For such complex systems only CNNs appears successful as it will be shown in this research study.

It is believed that CNN-based model structures offer quite a general framework for identifying nonlinear systems [38]. In the case-study presented in this chapter, the focus has been on a dynamic system with structurally unknown nonlinearities. Thus, even though the example considered does not represent any particular physical

system. no apriori knowledge concerning its structure is used in the identification process.

This chapter is organized as follows: The next section gives a detailed description of the polynomial NARX SI technique. Section IV.3 presents the case-study and all the relevant studies performed with it, within both the conventional and the CNN methods. The final section is a summary of all the lessons learned from the investigations on this simple case-study.

IV.2 The Polynomial Nonlinear AutoRegressive with eXogeneous Input Model Structure

A nonlinear regression model that is analogous to a linear regression, is a functional representation of nonlinear systems. Nonlinear regression models contain nonlinear terms in the regression vector, which are a combination of delayed inputs and outputs. For a discrete-time nonlinear system, the inputs and outputs belonging to finite dimensional vector spaces with dimensions r and m can be represented as follows:

$$\mathbf{u}(k) = [u_1(k) \ u_2(k) \ \cdots \ u_r(k)]^T, \quad (63)$$

$$\mathbf{y}(k) = [y_1(k) \ y_2(k) \ \cdots \ y_m(k)]^T. \quad (64)$$

In predictor form, the representation of the polynomial NARX model structure can be expressed as:

$$\begin{aligned} \hat{y}_i(k/k-1) = f_i[& y_1(k-1), \dots, y_1(k-n_y^1), y_2(k-1), \dots, y_2(k-n_y^2), \\ & \dots, y_m(k-1), \dots, y_m(k-n_y^m), \\ & u_1(k-1), \dots, u_1(k-n_u^1), u_2(k-1), \dots, u_2(k-n_u^2), \\ & \dots, u_r(k-1), \dots, u_r(k-n_u^r)] \end{aligned} \quad (65)$$

where $i = 1, 2, \dots, m$ and $f_i(\cdot)$ is some nonlinear function. For an equal number of delayed input and output vector elements used in each of the m output approximations, that is $n_y^1 = n_y^2 = \dots = n_y^m = n_y$ and $n_u^1 = n_u^2 = \dots = n_u^r = n_u$, the following predictor form for the vector NARX is obtained:

$$\hat{y}_i(k/k-1) = f_i[y(k-1), \dots, y(k-n_y), u(k-1), \dots, u(k-n_u)]. \quad (66)$$

Equation (66) is the multivariable NARX model structure in predictor form.

Billings and his colleagues were first to report the polynomial NARX nonlinear model structure [5], [10], [11], [12]. They have worked extensively in the area of input-output nonlinear SI using a number of methods. Examples of some of the nonlinear SI methods reported by Billings and his co-workers are the recursive input-output methods for multivariable discrete-time systems, the orthogonal least-squares methods, and the prediction error methods for stochastic systems. Billings has proposed the following polynomial parametrization for the NARX structure [11]:

$$\begin{aligned} y_i(k) = & \theta_0^{(i)} + \sum_{j_1=1}^n \sum_{j_2=j_1}^n \theta_{j_1 j_2}^{(i)} x_{j_1}(k) x_{j_2}(k) + \dots \\ & - \sum_{j_1=1}^n \dots \sum_{j_l=j_{l-1}}^n \theta_{j_1 \dots j_l}^{(i)} x_{j_1}(k) \dots x_{j_l}(k) + e_i(k), \quad i = 1, \dots, m, \end{aligned} \quad (67)$$

where l is the degree of the polynomial expansion and where

$$n = m \times n_y + r \times n_u, \quad (68)$$

suggesting that an equal number of delayed input and output vector components is used in the expansion. Furthermore, the following expansion terms are defined:

$$\begin{aligned} x_1(k) &= y_1(k-1), \quad x_2(k) = y_1(k-2), \quad \dots, \quad x_{m \times n_y}(k) = y_m(k-n_y), \\ x_{m \times n_y + 1}(k) &= u_1(k-1), \quad \dots, \quad x_n(k) = u_r(k-n_u). \end{aligned} \quad (69)$$

The polynomial expansion of equation (67) can be rewritten in the following regression form:

$$z(k) = \sum_{i=1}^M p_i(k) \theta_i + \xi(k), \quad \text{for } k = 1, \dots, N, \quad (70)$$

where N is the data length, $z(k) = y_i(k)$, $p_i(k)$ are monomials $x_1(k)$ to $x_n(k)$ up to degree l , $\xi(k)$ is the modeling error, and θ_i are unknown parameters to be estimated, and M is the total number of terms in the polynomial. Therefore, each of the m system outputs can be independently approximated using the polynomial expansion of equation (67). Thus, the identification of an r -input and m -output system can be reduced to the identification of m , r -input single-output systems. In the following paragraph an estimation technique for the polynomial NARX is presented which has been proposed by Billings [13].

The orthogonal least-squares method is used in nonlinear SI because of the following reasons:

- (1) Parameter estimation methods using least-squares estimation are well developed and applicable to the polynomial NARX model structure [13];
- (2) A nonlinear system can be expressed as a polynomial expansion, in which, the parameters are linear. Equation (70) can now be written in the following matrix form:

$$\mathbf{z} = \mathbf{P}\boldsymbol{\Theta} + \boldsymbol{\Xi}, \quad (71)$$

where

$$\mathbf{z} = [z(1) \ z(2) \ \dots \ z(N)]^T, \quad (72)$$

$$\mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_M], \quad (73)$$

$$\Theta = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_M]^T, \quad (74)$$

$$\Xi = [\xi(1) \quad \xi(2) \quad \cdots \quad \xi(N)]^T, \quad (75)$$

and where

$$\mathbf{p}_i = [p_i(1) \quad p_i(2) \quad \cdots \quad p_i(N)]^T, \quad \text{for } i = 1, \dots, M. \quad (76)$$

Thus the matrix \mathbf{P} is $N \times M$ dimensional. A linear least-squares estimation algorithm can now be applied to obtain the parameters $\hat{\Theta}$ which minimize $\|\mathbf{z} - \mathbf{P}\Theta\|$, where $\|\cdot\|$ is the Euclidian norm. It is well-known that the solution $\hat{\Theta}$ will satisfy the following equation [13]:

$$\mathbf{P}^T \mathbf{P} \Theta = \mathbf{P}^T \mathbf{z}, \quad (77)$$

where $\mathbf{P}^T \mathbf{P}$ is called the information matrix.

A full model using equation (70) can easily involve an excessive number of terms. For example, let $m = r = 2$, the maximum lags be 2, and the polynomial degree be 2, then the number of expansion terms is 182. If $m = r = 10$, the maximum lags be 12, and the polynomial degree be 4, then the number of parameters will result in a very large matrix, leading to an ill-conditioned problem. The determination of the structure, or which terms to include in the model from the large number of candidate terms, is therefore essential in multivariable nonlinear SI.

In the following, we consider the combined problem of structure selection and parameter estimation, based on the orthogonal decomposition of the regression matrix \mathbf{P} , using the classical Gram-Schmidt method. A simple and efficient algorithm can be

derived that determines P_s , a subset of P , in a forward-regression manner by choosing one column of P at a time for which the sum of squares of residuals is maximally reduced [16]. Assume that P_s has M_s ($M_s < M$ and $M_s \leq N$) columns. Factorize P_s into $W_s A_s$ as follows:

$$P_s = W_s A_s, \quad (78)$$

note that

$$g_s = A_s \Theta_s, \quad (79)$$

where

$$A_s = \begin{bmatrix} 1 & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1M} \\ 0 & 1 & \alpha_{23} & \dots & \alpha_{2M} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & \alpha_{(M-1)M} \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}. \quad (80)$$

A_s is an $M_s \times M$ unit upper triangular matrix and

$$W_s = [w_1 \quad \dots \quad w_{M_s}], \quad (81)$$

is an $N \times M_s$ matrix with M_s orthogonal columns that satisfies:

$$W_s^T W_s = D_s, \quad (82)$$

and where D_s is a positive diagonal matrix that enters the decomposition:

$$P_s^T P_s = A_s^T W_s^T W_s A_s = A_s^T D_s A_s. \quad (83)$$

The residuals are now defined as follows:

$$\hat{\Xi} = z - P_s \hat{\Theta}_s = z - (P_s A_s^{-1})(A_s \hat{\Theta}_s) = z - W_s g_s, \quad (84)$$

where

$$\hat{\Xi} = [\hat{\xi}(1) \quad \dots \quad \hat{\xi}(N)]^T. \quad (85)$$

Rewriting equation (84) as follows:

$$z = W_s g_s + \hat{\Xi}. \quad (86)$$

The results in the following sum of squares of the variable z

$$\langle z, z \rangle = \sum_{k=1}^N z^2(k) = \sum_{i=1}^{M_s} g_i^2 \langle w_i, w_i \rangle + \langle \hat{\Xi}, \hat{\Xi} \rangle. \quad (87)$$

The error reduction ratio due to w_i is thus defined as the proportion of the dependent variable variance explained by w_i , as follows:

$$[\text{err}]_i = \frac{g_i^2 \langle w_i, w_i \rangle}{\langle z, z \rangle}. \quad (88)$$

Using the classical Gram-Schmidt algorithm, the parameters of this model structure can be obtained by the following procedure:

- (1) Denote $w_1^i = p_i$ for $i = 1, 2, \dots, M$ and compute

$$g_1 = \frac{\langle w_1^1, z \rangle}{\langle w_1^1, w_1^1 \rangle}, \quad [\text{err}]_1^1 = \frac{(g_1^1)^2 \langle w_1^1, w_1^1 \rangle}{\langle z, z \rangle}. \quad (89)$$

Assume that $[\text{err}]_1^1 = \max\{[\text{err}]_1^j, 1 \leq j \leq M\}$. Then $w_1 = w_1^1 (= p_1)$ is selected as the first column of W_s , the first element of g_s , $g_1 = g_1^1$, and $[\text{err}]_1 = [\text{err}]_1^1$.

- (2) At the next step, for $i = 1, \dots, M$ and $i \neq j$, compute:

$$\begin{aligned} \alpha_{12}^i &= \frac{\langle w_1, p_i \rangle}{\langle w_1, w_1 \rangle}, & w_2^i &= p_i - \alpha_{12}^i w_1, \\ g_2^i &= \frac{\langle w_2^i, z \rangle}{\langle w_2^i, w_2^i \rangle}, & [\text{err}]_2^i &= \frac{(g_2^i)^2 \langle w_2^i, w_2^i \rangle}{\langle z, z \rangle}. \end{aligned} \quad (90)$$

Assume that $[\text{err}]_2^k = \max\{[\text{err}]_2^i, 1 \leq i \leq M \text{ and } i \neq j\}$. Then $w_2 = w_2^k (= p_k - \alpha_{12}^k w_1)$ is selected as the second column of W_s , together with the second column of A_s , $\alpha_{12} = \alpha_{12}^k$, the second element of g_s , $g_2 = g_2^k$, and $[\text{err}]_2 = [\text{err}]_2^k$.

(3) The selection procedure is continued until the M_s th for which when

$$1 - \sum_{i=1}^{M_s} [\text{err}]_i < \rho, \quad (91)$$

where ρ ($0 < \rho \leq 1$) is the desired tolerance. Since the matrix A_s is an upper triangular matrix, the parameters $\hat{\Theta}_s$ can be easily computed from equation (79) by backward substitution. The value of ρ determines how many terms ($= M_s$) will be included in the final model of system. There is, however, a problem. The value of $[\text{err}]_i$ may depend upon the order in which the term $p_i(t)$ enters the equation. As a result, simply orthogonalizing the $p_i(k)$'s into the orthogonal equation in the order in which they happen to be written down in equation (70), may produce the wrong information regarding their significance as determined by the criterion of equation (91). This difficulty can, however, be overcome by the forward-regression procedure proposed by Billings et al. [3]. Numerical simulations that use this forward-regression algorithm are given in the next section.

For the Nonlinear AutoRegressive Moving Average with eXogeneous Input Model structure (NARMAX), delayed noise terms, $e(t - j)$, are included in the model and the properties of these terms have to be estimated using the prediction errors or the residuals, $\xi(t - j)$. The orthogonal property of the estimator ensures that the selection of the process noise model parameters can be decoupled. Significant terms in each row of the process model are first selected. Initial residuals are then computed based on the process models and the assumed model structure. The residuals are treated as extra input to the system. Based on these residuals, a revised residual sequence is calculated and an improved noise model is determined. This process allows

the model to determine the colored noise structure associated with the operation of the system. However, in complex process systems with high process and sensor noise, the polynomial NARMAX technique appears to give no better results than the polynomial NARX technique. Further information on the polynomial NARMAX technique is provided in the literature [3].

IV.3 A Case-Study

In the example presented in this section both the conventional and the CNN methods for SI will be examined and compared. The conventional method utilized is the polynomial NARX technique, as it is considered the most powerful method in conventional nonlinear SI. Many CNN methods and variations will be investigated in detail, such as batch update vs. individual update, teacher forcing (TF) vs. global feedback (GF), and weight decay. CNN methods will be also combined with the polynomial NARX technique in new, innovative approaches to be presented. Note that the objective is primarily to design a CNN model capable of performing accurate multi-step-ahead prediction (MSP), and not single-step-ahead prediction (SSP). Conventional identification methods are capable of performing this latter task even in complex systems, however, CNN methods give relatively better results. The validation tests to be performed will be for both the MSP and the SSP. The reason behind testing for SSP is to make sure that the model still performs well for SSP although it was designed to perform MSP. A model capable of performing both MSP and SSP is much more reliable than a model capable of performing SSP only. The former

demonstrates that the model has learned the dynamics of the system, while the latter does not. Since the model will be desired to perform equally well for SSP and MSP, it may not be as good as a model performing well only in one task, namely the SSP. The validation tests to be presented will show both, the SSP and the MSP. Throughout this research study a CRAY-2 YMP supercomputer along with some powerful Sun Spark Stations were used to conduct all the required investigations.

Throughout this chapter the error criterion adopted is the relative mean-squared-error (REMSE) given by the following equation:

$$\text{Error}(i) \equiv \frac{\sum_{k=1}^N (\hat{y}_i(k/k-1) - y_i(k))^2}{\sum_{k=1}^N (y_i(k) - \bar{y}_i)^2}; i = 1, 2, \quad (92)$$

where N is the number of data points considered, $y_i(k)$ is the value of the observed output, $\hat{y}_i(k/k-1)$ is the value of the predicted output, and \bar{y}_i is the mean value of the observed output.

The stochastic MIMO nonlinear system used in this case-study can be expressed by the following state and output equations:

$$x_1(k) = 0.5(x_1^2(k-1))^{\frac{1}{3}} + 0.3x_2(k-1)x_3(k-1) + 0.2u_1(k-1) + n_{x_1}^{process}(k), \quad (93)$$

$$x_2(k) = 0.5(x_2^2(k-1))^{\frac{1}{3}} + 0.3x_3(k-1)x_1(k-1) + 0.5u_1(k-1) + n_{x_2}^{process}(k). \quad (94)$$

$$x_3(k) = 0.5(x_3^2(k-1))^{\frac{1}{3}} + 0.3x_1(k-1)x_2(k-1) + 0.5u_2(k-1) + n_{x_3}^{process}(k), \quad (95)$$

and

$$y_1(k) = 0.5(x_1(k) + x_2(k) + x_3(k)), \quad (96)$$

$$y_2(k) = 2(x_1^2(k))^2, \quad (97)$$

where $n_{x_i}^{process}(k)$ for $i = 1, 2, 3$, is the process noise of the system.

The system is a two input two output (2I2O), with three dynamic states corrupted by process noise. The training data set consisted of all possible 25 combinations of step inputs with magnitudes 0.125, 0.25, 0.375, 0.5, as well as the zero input for both input channels. Each signal consists of 15 samples. Five different pulses of suitable shapes, each containing 40 samples, were added to the training data set, for a total of 575 samples. The process noise is assumed zero mean, white Gaussian noise with 0.05 standard deviation.

Three types of tests are performed with entirely new signals for investigating the models' predictive performance. The first test signal is $u_1(k) = 0.3 + 0.2\sin(\frac{\pi k}{8})$, $u_2(k) = 0.2$, where the step input in the second channel is delayed by 5 time steps with respect to the input in the first channel. The second test set consists of a step augmented with zero mean, white Gaussian noise. The magnitudes for the steps are 0.3 and 0.2 for the two input channels, respectively. The third test consists of pulses given by $u_1(k) = 0.3$ for $k \leq 25$ and $u_2(k) = 0.3$ for $k \in [10, 35]$. Each test signal is augmented with zero mean, white Gaussian noise with 0.05 standard deviation, simulating low noise environment, and 0.08 standard deviation, simulating high noise environment. In the following subsections results using the polynomial

NARX technique and the results of the investigations using CNNs, will be presented in detail.

IV.3.1 Application of the Polynomial NARX Method

A polynomial NARX model structure was obtained using the forward-regression orthogonal method. A detailed presentation of this technique was given in section IV.2. Many models were obtained using this technique. These models were obtained by choosing different numbers of delayed input(s)/outputs(s) and order of the polynomial. Only the model which performed best is presented. The obtained model is as follows:

$$\begin{aligned}\hat{y}_1(k/k-1) = & 0.79y_1(k-1) + 0.3u_1(k-1) - 0.43y_1(k-3)u_1(k-2) \\ & + 1.3u_1^2(k-2)u_2(k-1) + 0.17y_1(k-3),\end{aligned}\tag{98}$$

$$\begin{aligned}\hat{y}_2(k/k-1) = & 0.69y_2(k-1) + 0.44y_1(k-1)u_1(k-1) + .04y_1(k-4) \\ & + 1.08u_1^2(k-3)u_2(k-3) - 0.66u_1(k-4)u_1(k-2)y_2(k-2).\end{aligned}\tag{99}$$

Figures 11 through 14 give the SSP of the two system outputs for the three tests chosen as validation sets for both low and high noise levels. Similarly, Figures 15 through 18 give the MSP for the same two outputs. The REMSE for each of these validation tests is summarized in Table 1. The errors displayed are the average errors of output one and output two. This table shows that the errors for the MSP are higher than the errors of the SSP. The results of this table will be updated later with the results from CNN methods, for comparasion purposes. Even though, the polynomial

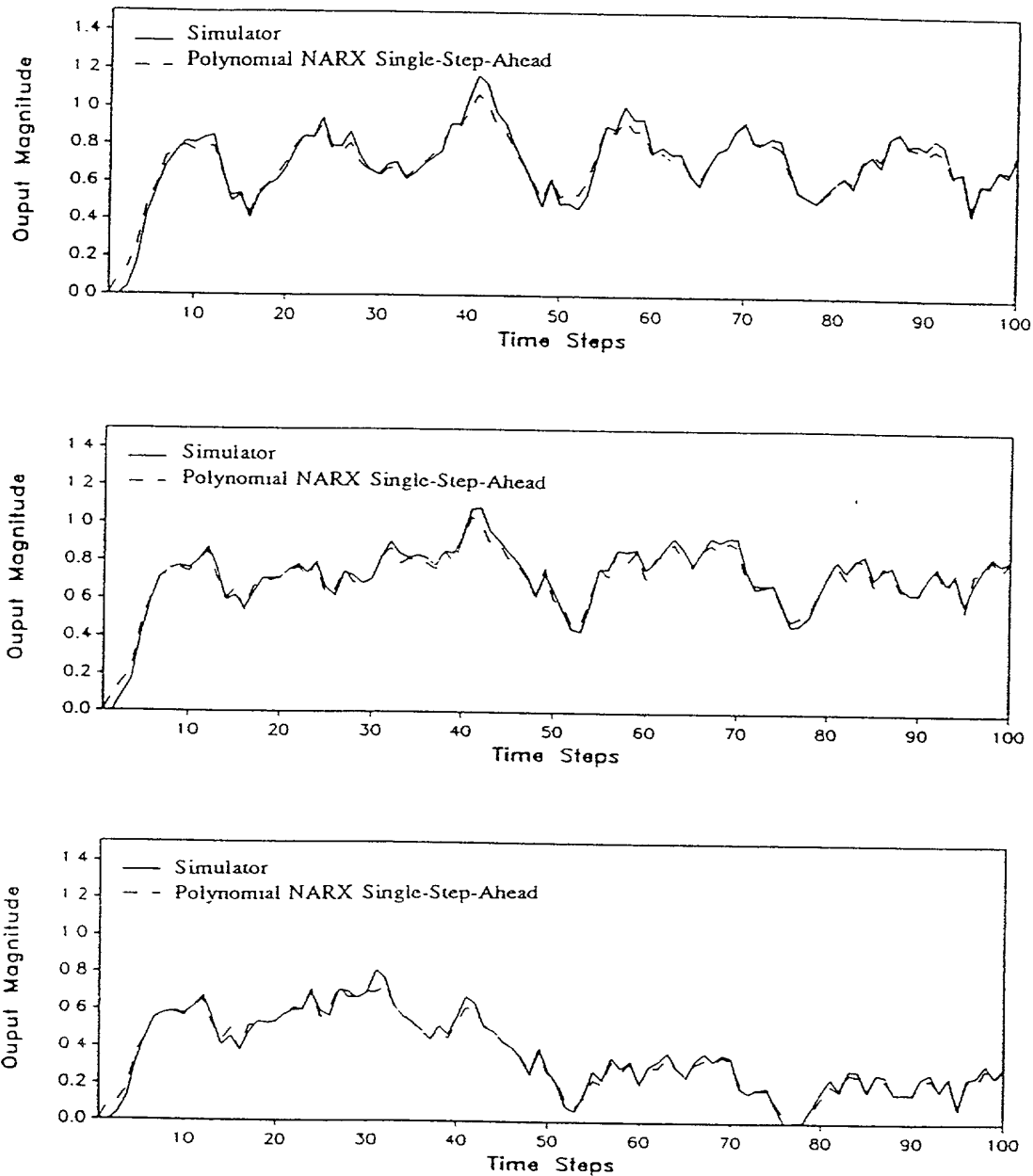


Figure 11. First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

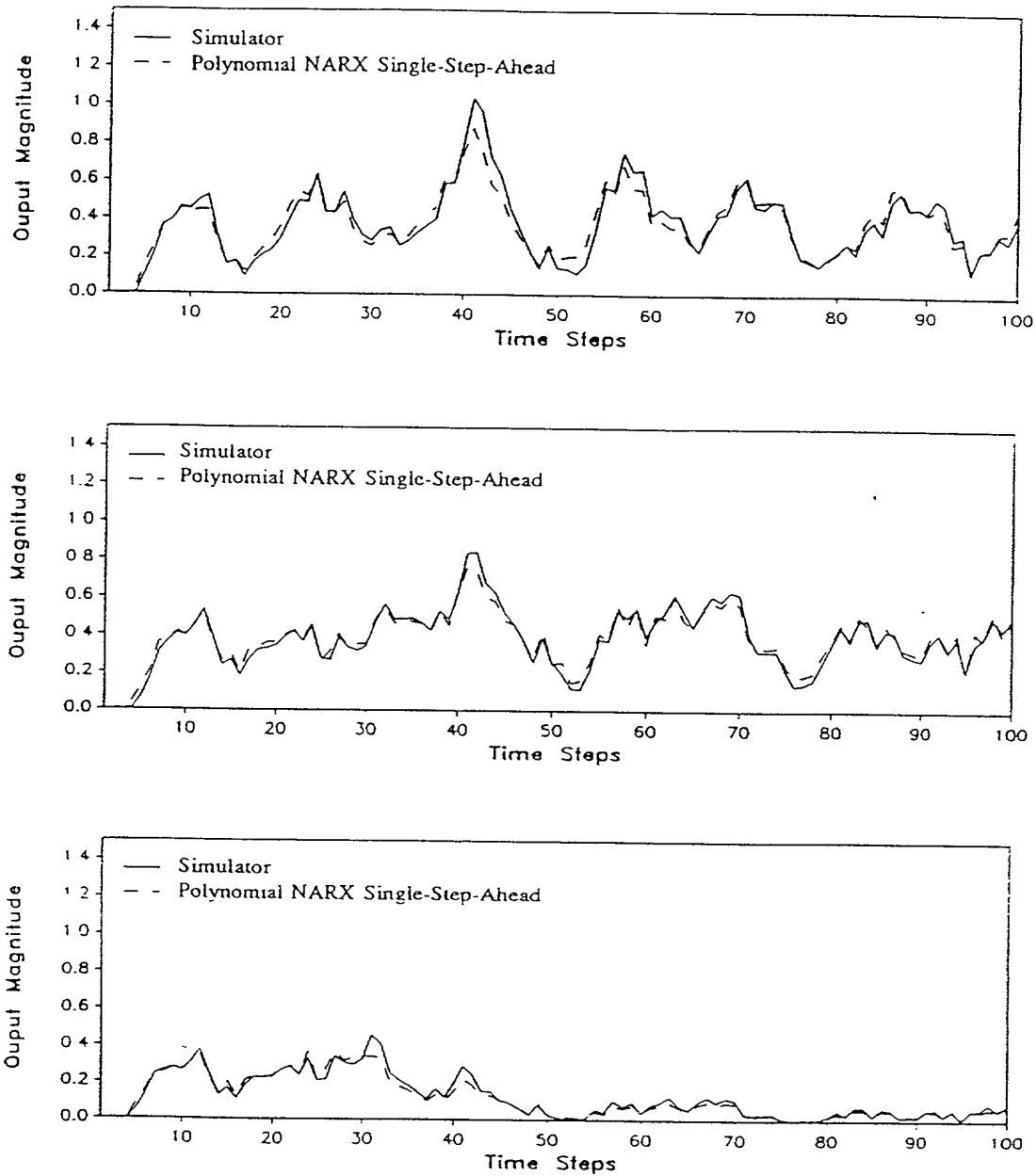


Figure 12. Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

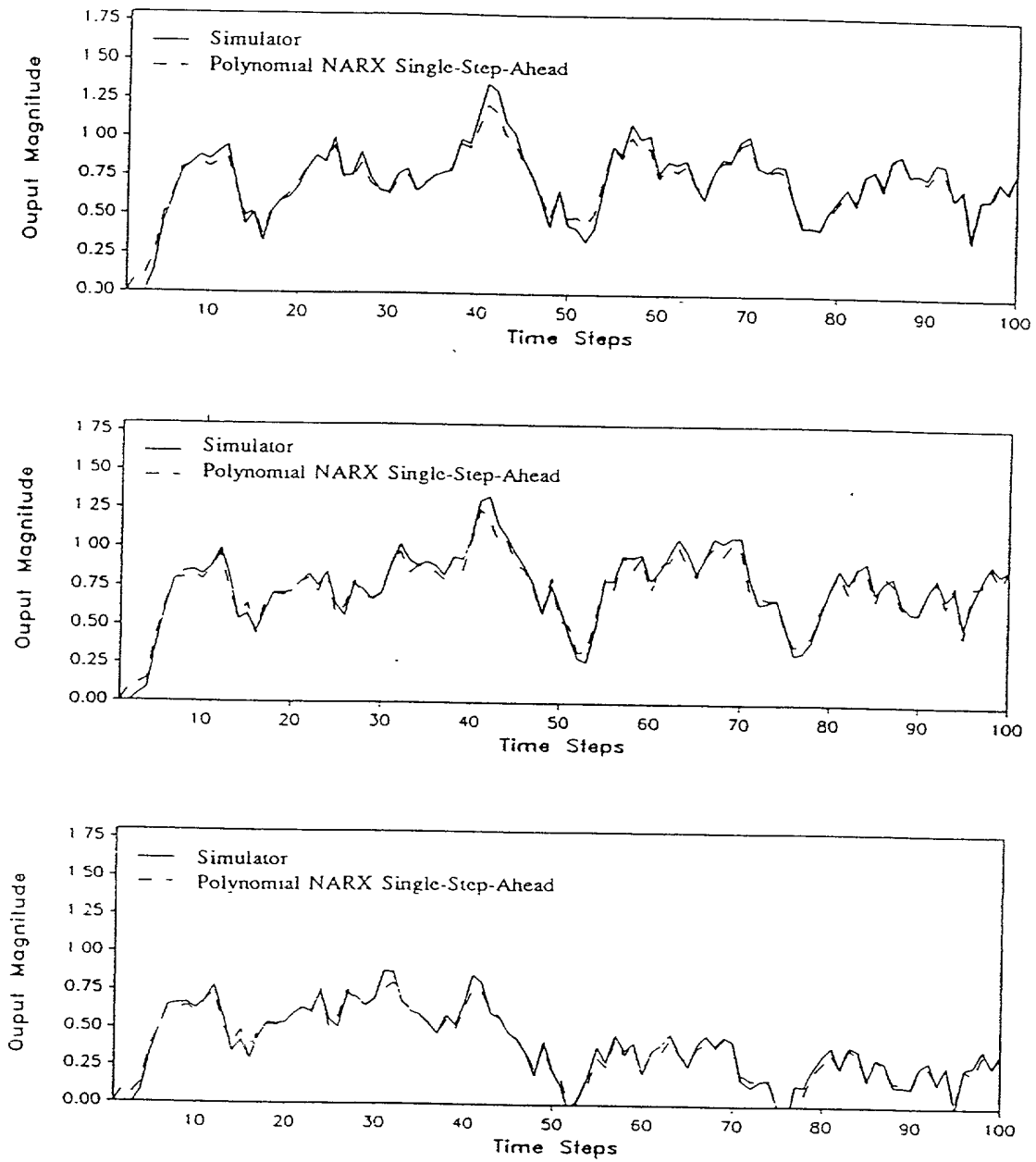


Figure 13. First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

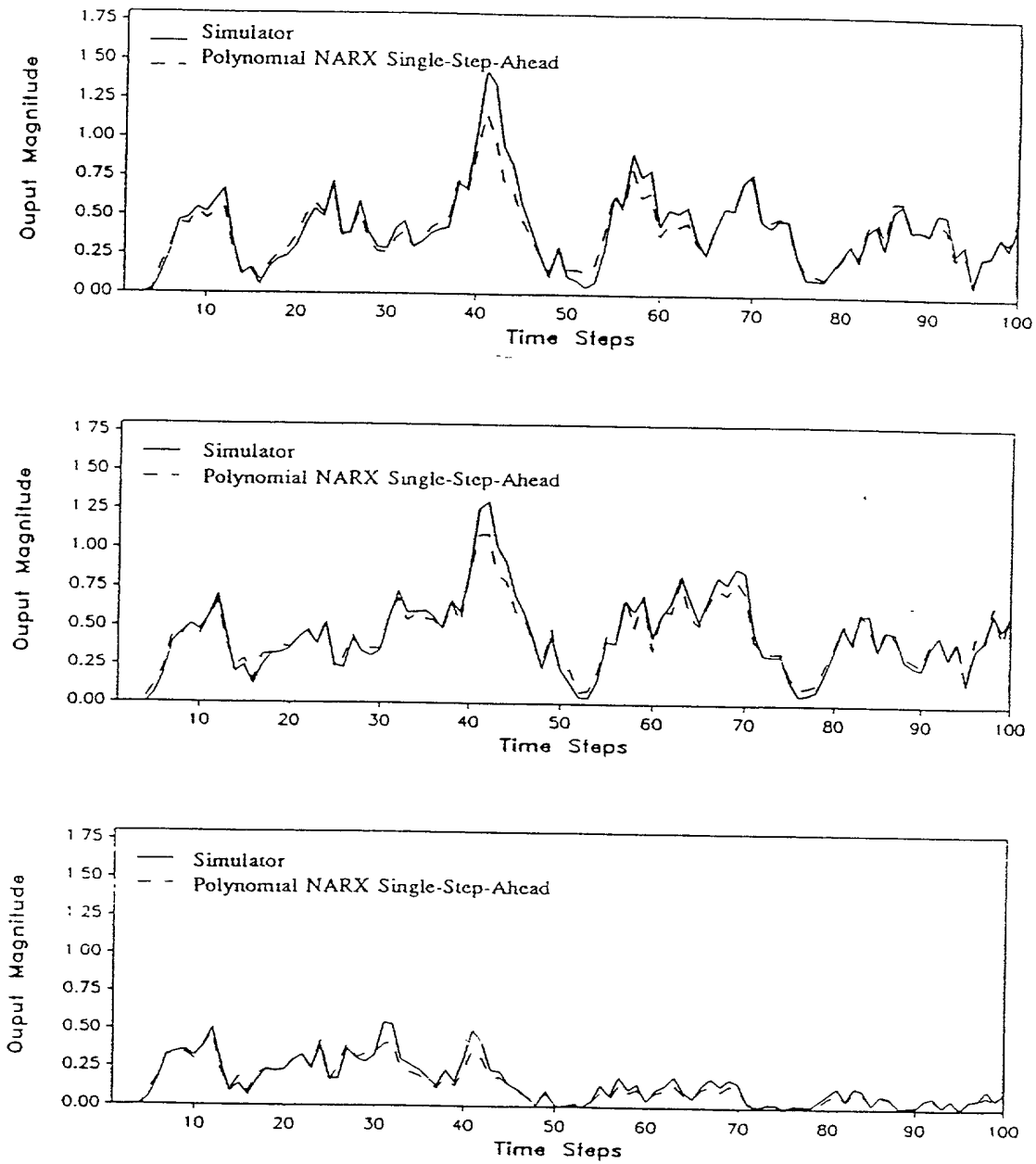


Figure 14. Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

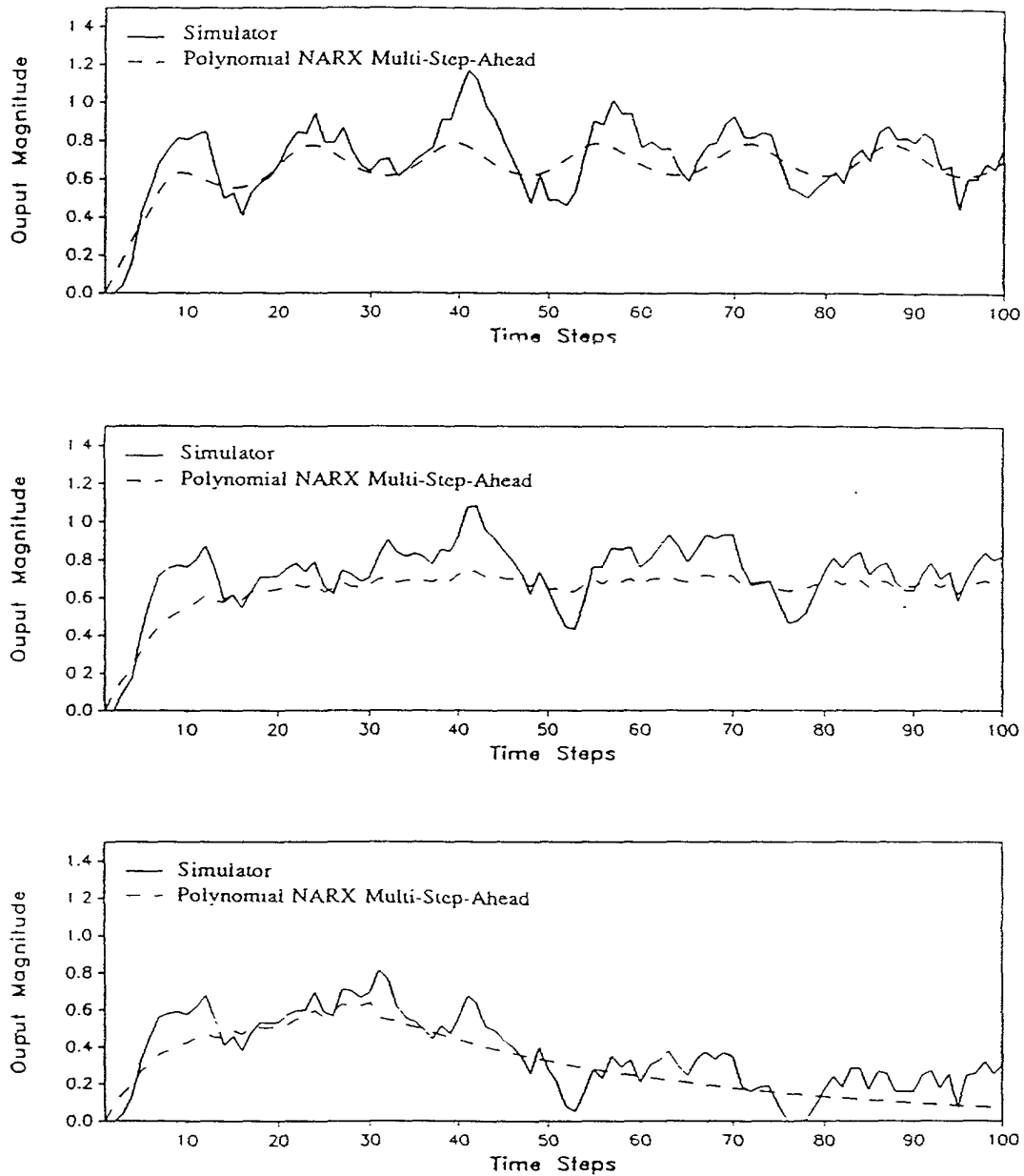


Figure 15. First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

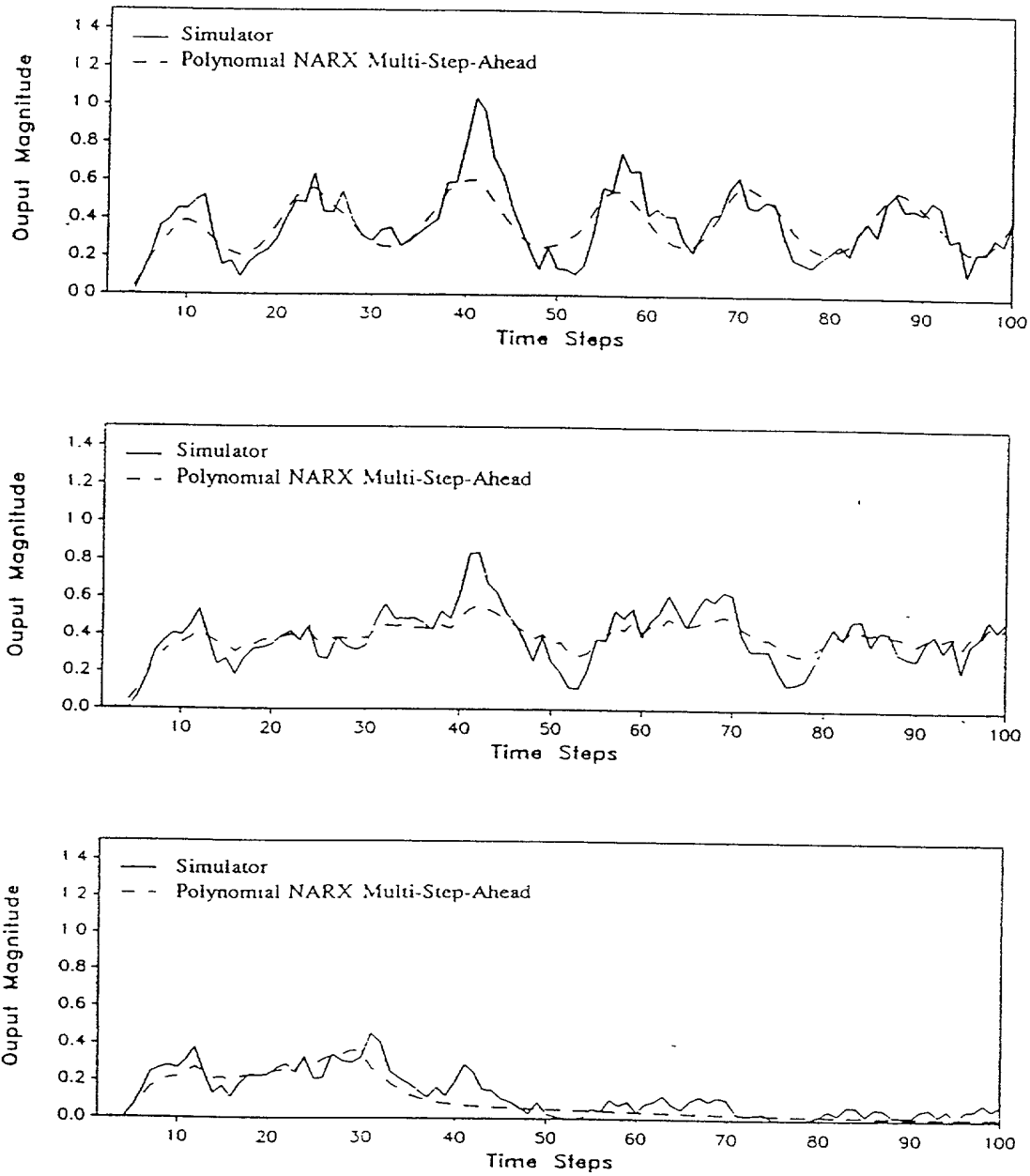


Figure 16. Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

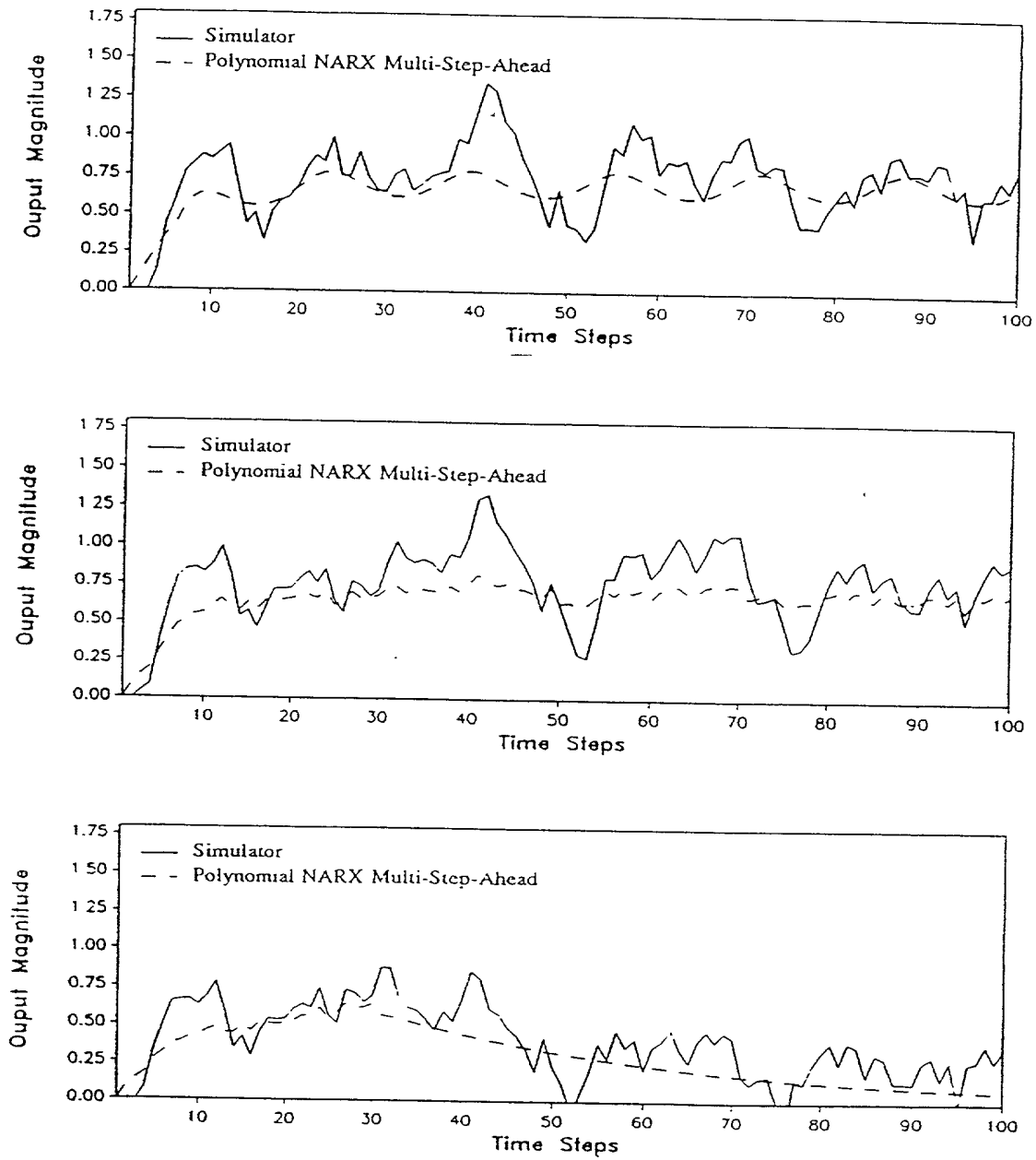


Figure 17. First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input : Bottom: Pulse Input.)

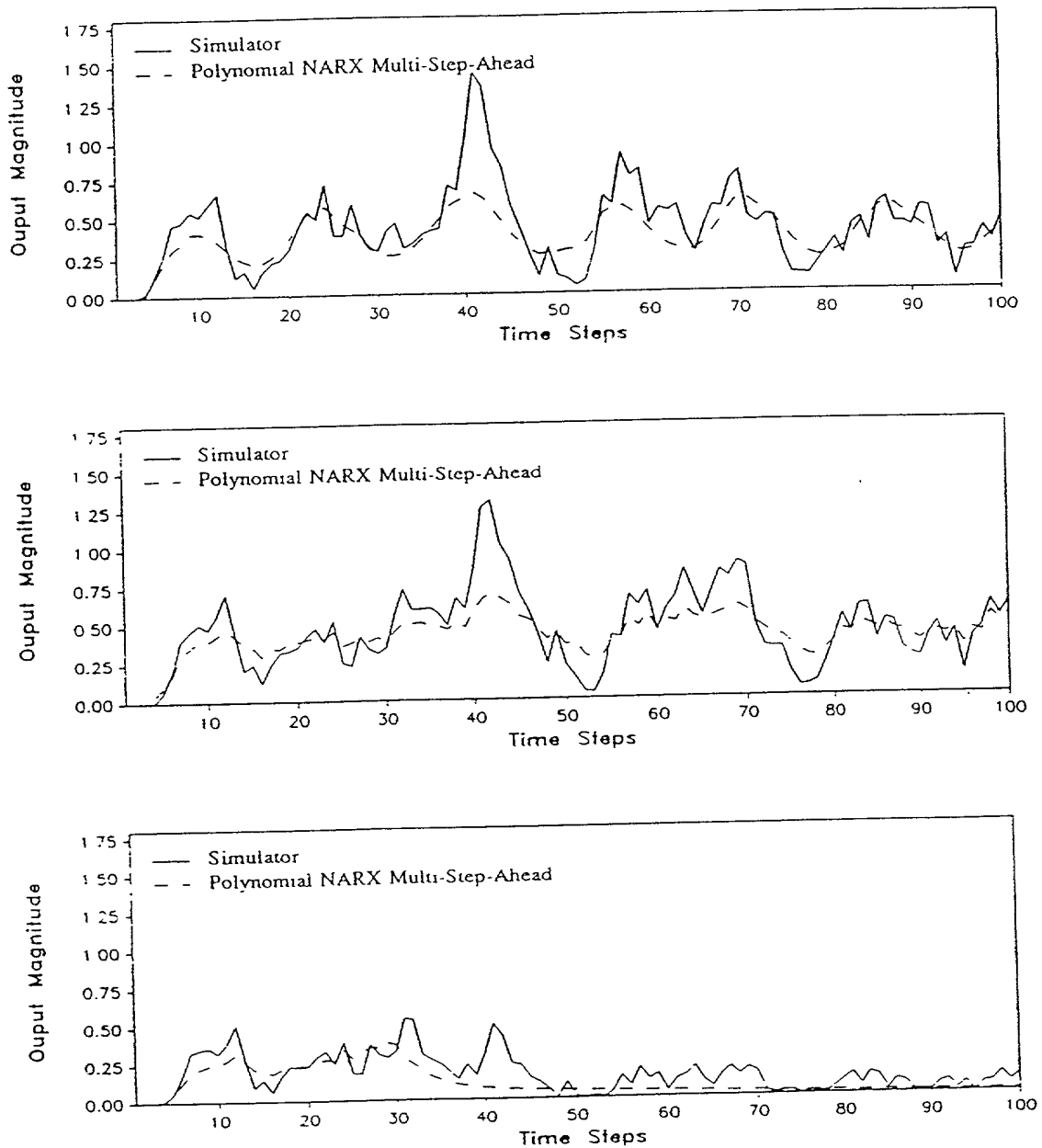


Figure 18. Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

Table 1. Relative Mean-Squared Errors for the Case-Study using the Polynomial NARX.

Model Structure	Sinusoidal Input	Step Input	Pulse Input
NARX SSP with $\sigma = 0.05$	0.19%	0.24%	0.17%
NARX SPP with $\sigma = 0.08$	0.30%	0.28%	0.33%
NARX MSP with $\sigma = 0.05$	0.39%	0.59%	0.36%
NARX MSP with $\sigma = 0.08$	0.60%	0.67%	0.66%

NARX technique is giving acceptable performance in MSP, we would still investigate CNN models because of their potential usefulness in more complex system, like the U-Tube Steam Generator (UTSG).

IV.3.2 Application of CNN Methods

Application of CNN methods require two data sets, the training data set and the testing data set. The training data set is the same as used in the polynomial NARX methods, comprised of 575 data points. The inputs of the testing data set are given by: $u_1(k) = .0045k - 0.2 * \sin(\pi(k-1)/6.0) + n^{process}(k)$, which is the summation of a ramp, a sinusoidal input, and some process noise: $u_2(k) = 0.2 + n^{process}(k)$, which is a step with some process noise. The total number of data points in the testing set is 100. The process noise is zero mean, white Gaussian, with 0.05 standard deviation. The validation data sets are chosen to be the same as the ones used for testing the polynomial NARX model, for the purpose of a fair comparison between the conventional SI method and the CNN methods. Note that in using the CNN method, the network was trained using the training data set and at each weight update the

error on the testing data set was computed. The best weights are chosen to be the ones which give the lowest error in the testing data set. Also note that the network architecture design is done according to the guidelines given in chapter III.

In using CNNs, many options were examined: These include using delayed inputs and/or delayed outputs in the input layer, batch weight update vs. individual weight update, teacher forcing vs. GF, weight decay, and use of the model structure obtained from the polynomial NARX structure selection algorithm as input layer for the CNN. Each of these options will be explained and presented in details in the following subsections. However, the quantitative results of only the successful options will be presented.

IV.3.2.1 Use of Delayed Outputs and/or Inputs in the Network Input Layer

The issue of using delayed input(s) and/or delayed output (s) in the network input layer was discussed in chapter III, within the context of network complexity. As it was stated earlier, the specific number of delayed inputs and outputs to be used is system dependent. In the current 2I2O artificial system, we have no way of telling which delayed outputs/inputs should be utilized. As a result, attempts have been made to utilize up to seven delayed outputs for each CNN. For each additional delayed output utilized, a network was performed to find the number of hidden nodes which results in the lowest error in the testing test. The search was conducted using TF in the training set, i.e. utilizing the observed delayed outputs, with the individual update mode. However, when computing the error in the testing test GF, (i.e. utilizing the predicted delayed outputs), was used ; As the objective has been to design a CNN

model capable of performing mutli-step ahead prediction. The learning rate used is $\eta = 0.001$.

In the first stage of the network search, two hidden layers were used with up to twelve nodes in each layer. This caused overfitting, and consequently the number of hidden layers was reduced to one with no more that four nodes in the hidden layer. This shift in network complexity during the design was a major break-through, because less time was consumed to perform a search for the desired network. Figure 19 shows the MSE of the testing set using GF, for the different number of delayed outputs utilized in the input layer of the network. Note that for each additional delayed output utilized, a node search was performed. Figure 19 presents the network with the best performance for it each case. As it can be seen from this figure, all of the networks converged to their lowest error within 10,000 iterations. This lowest error appears to be the threshold of overfitting. Also note that the lowest errors are very close to each other. This means that for this 2I2O artificial problem, utilizing more than one delayed output does not help the network improve its performance in generalization. The network which gives the lowest MSE error on the testing data set is the (2-3-2) network with one delayed output utilized. This network is used to generate the results presented in Figure 20 through Figure 27.

Figures 20 through 23 give the SSP of the system outputs to the three tests chosen as validation sets for both low and high noise levels, while Figures 24 through 27 give the MSP for the same outputs. The relative REMSE for each of these tests is summarized in Table 2. The errors displayed are the average errors of output one and

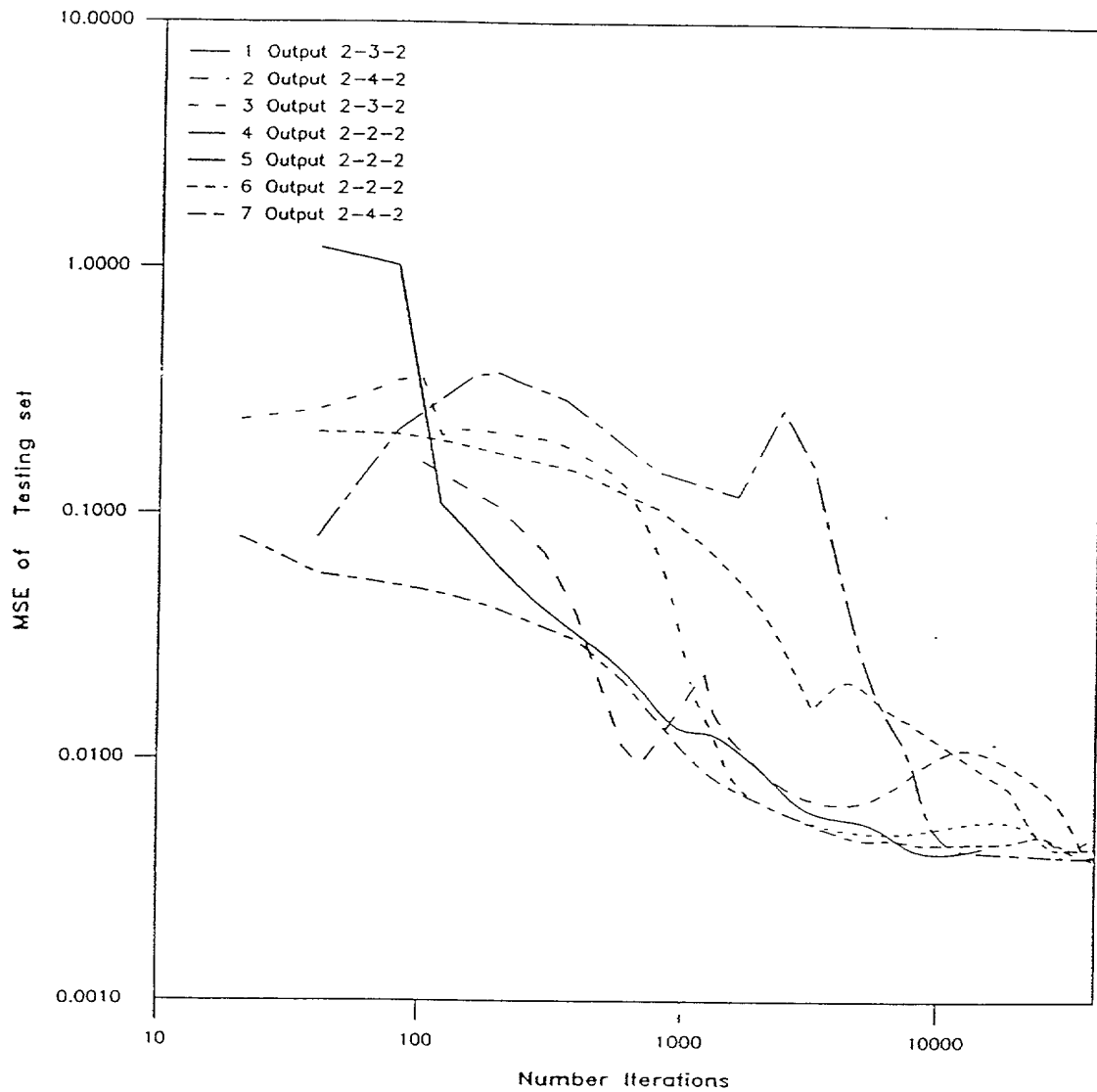


Figure 19. Mean Square Error of the Testing Set Using Global Feedback for Different Number of Delayed Outputs utilized in the Input Layer: Individual weight Update Mode.

164

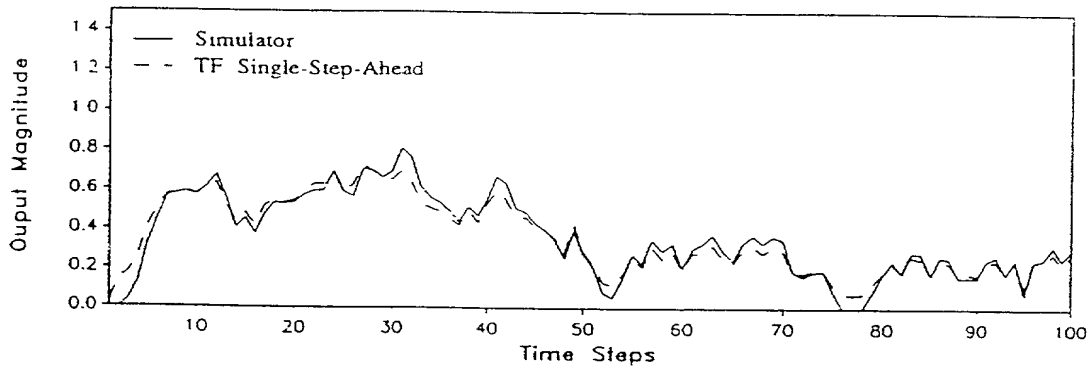
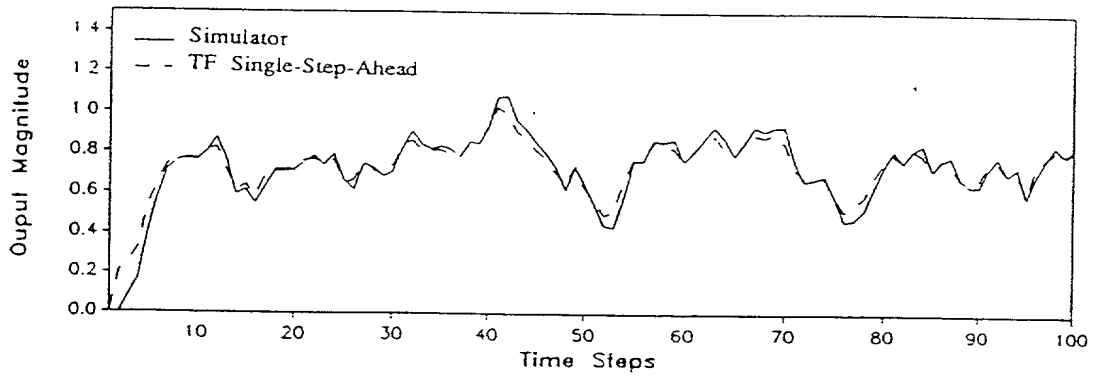
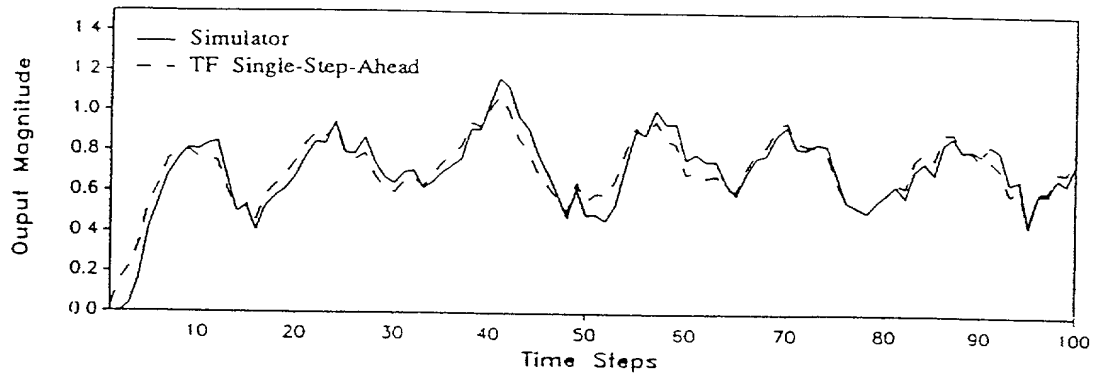


Figure 20. First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

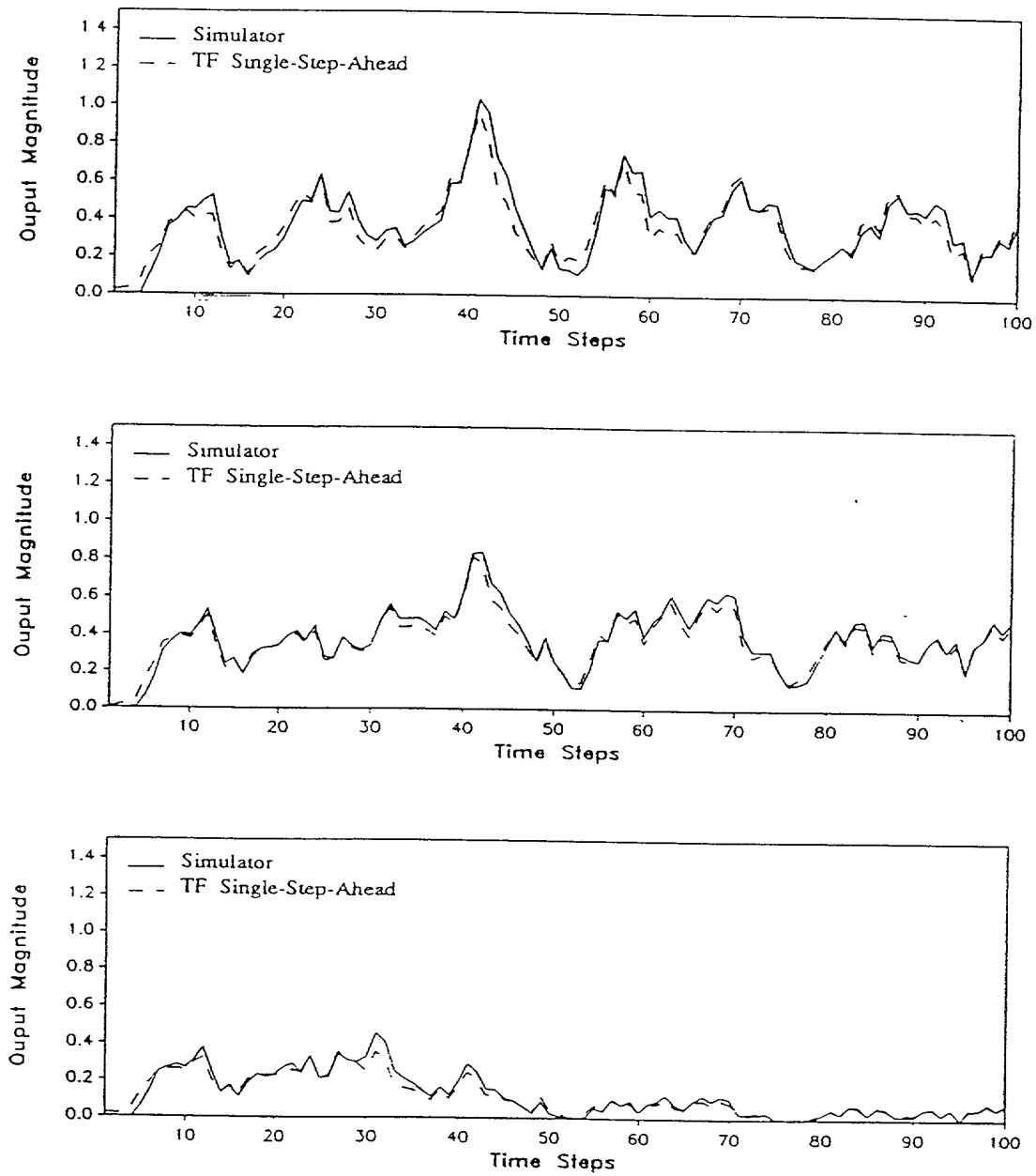


Figure 21. Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input : Bottom: Pulse Input.)

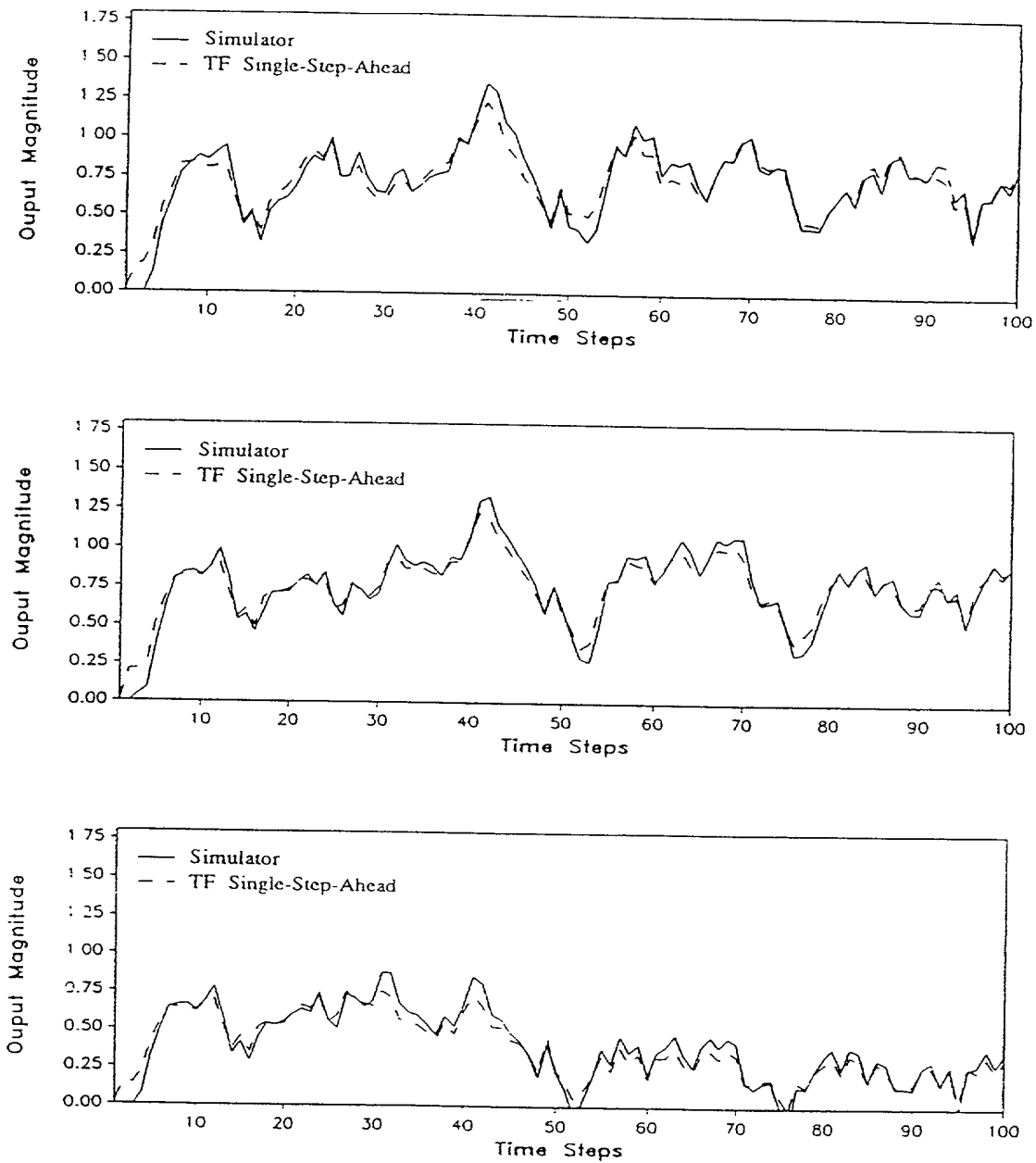


Figure 22. First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

167

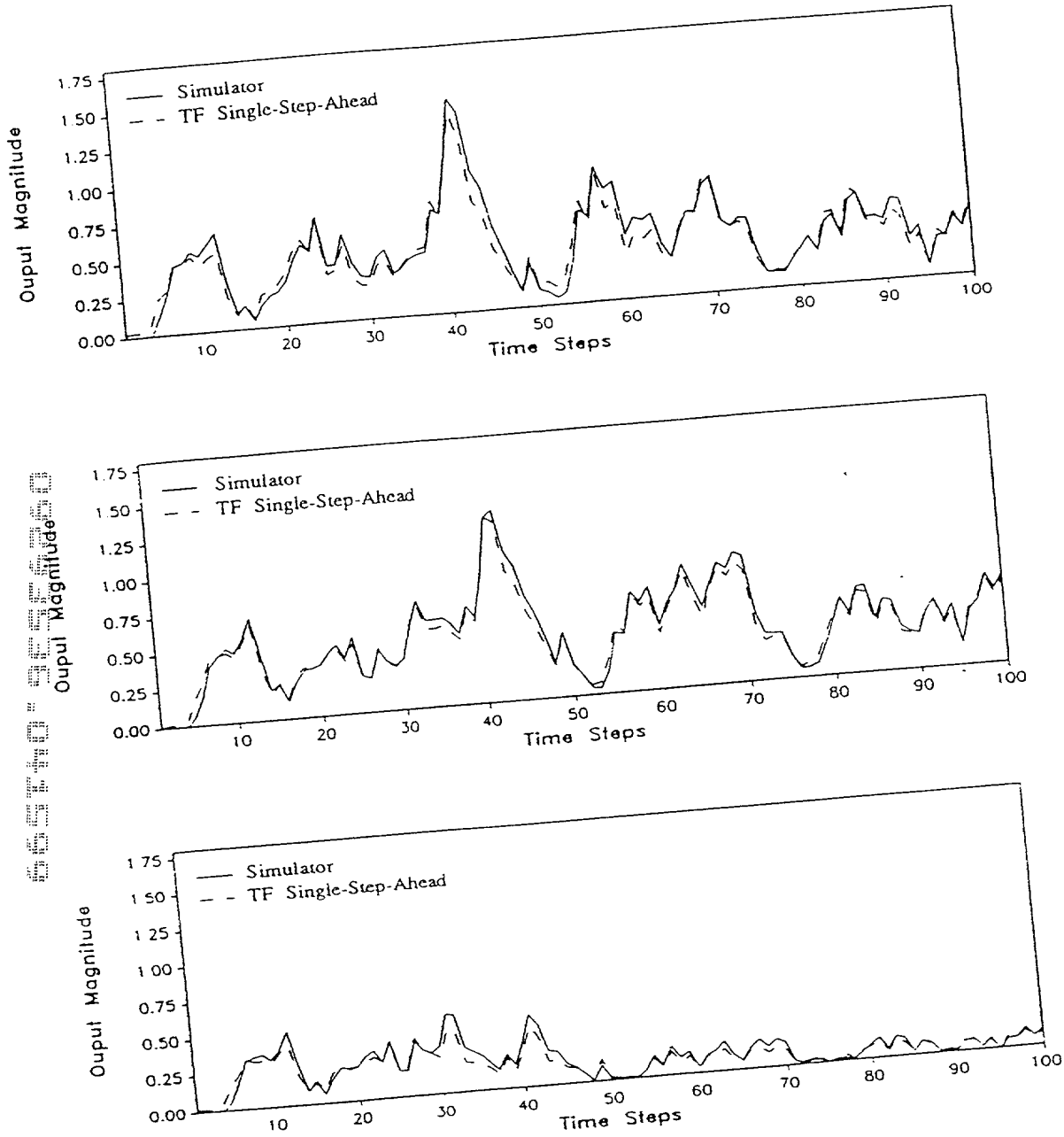


Figure 23. Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

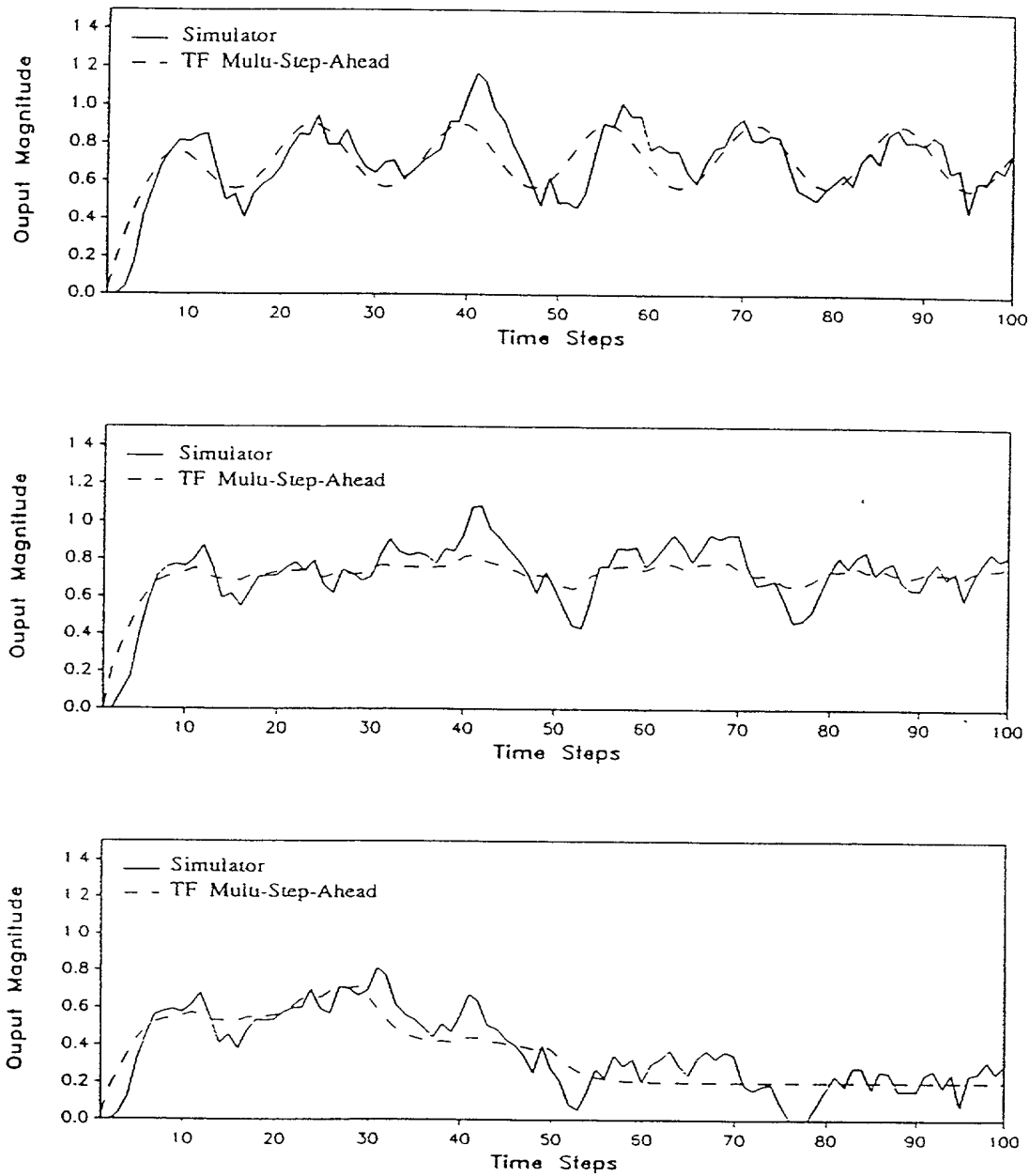


Figure 24. First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

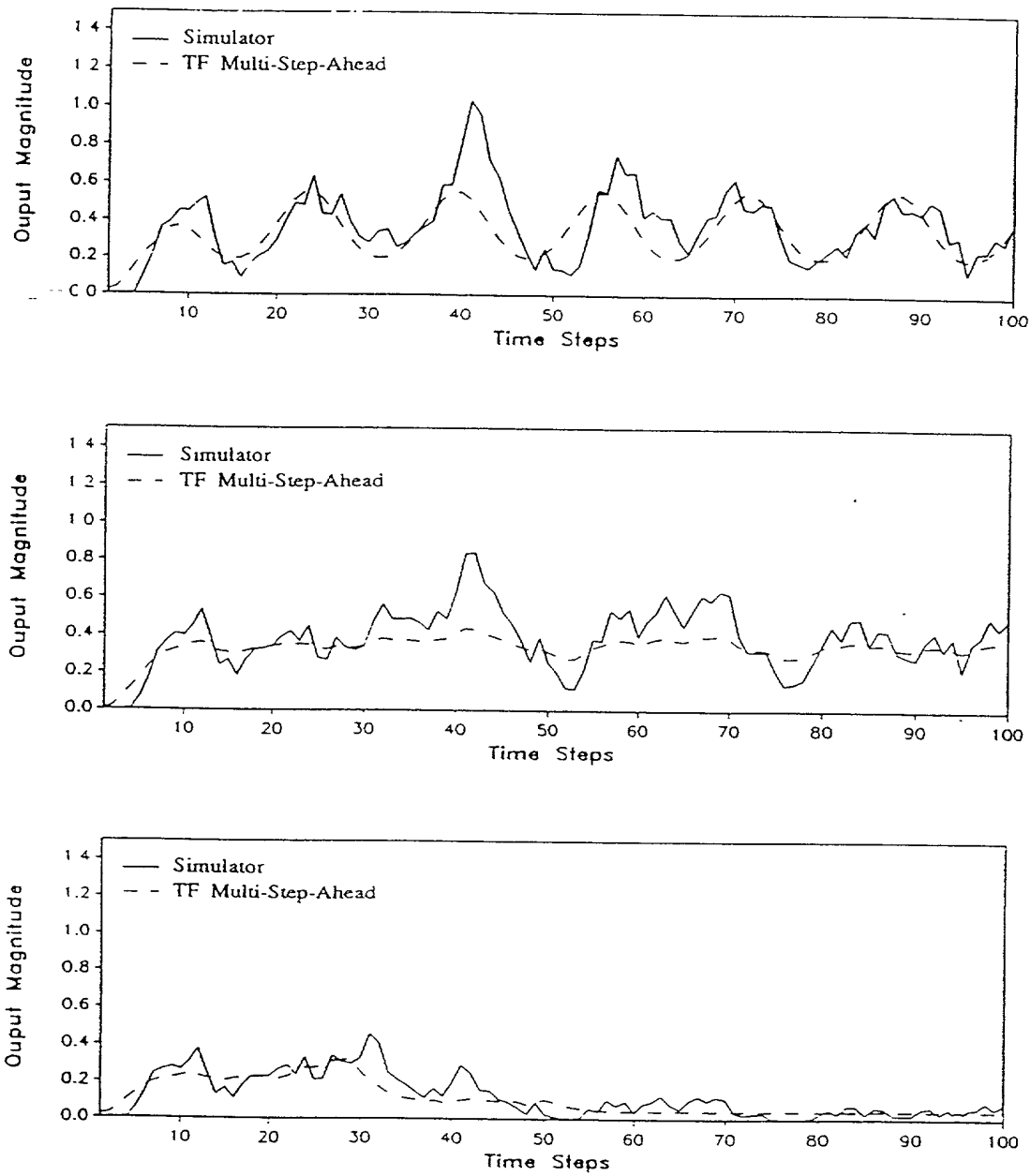


Figure 25. Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input : Bottom: Pulse Input.)

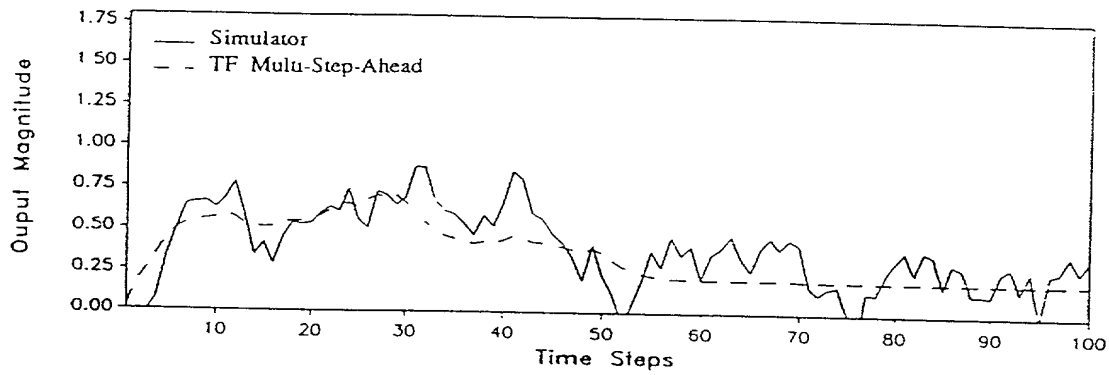
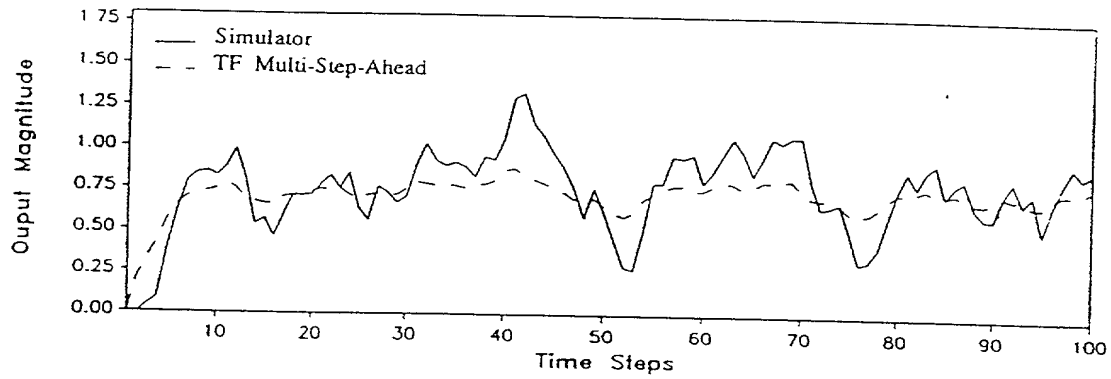
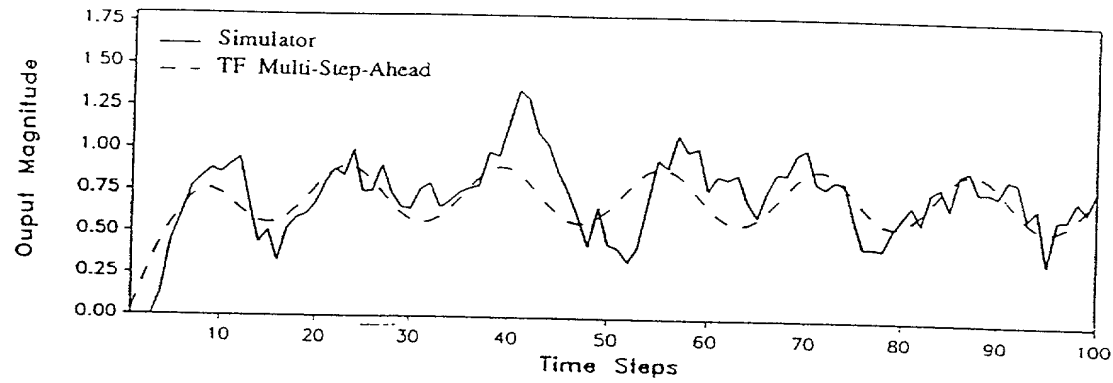


Figure 26. First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input, Middle: Step Input : Bottom: Pulse Input.)

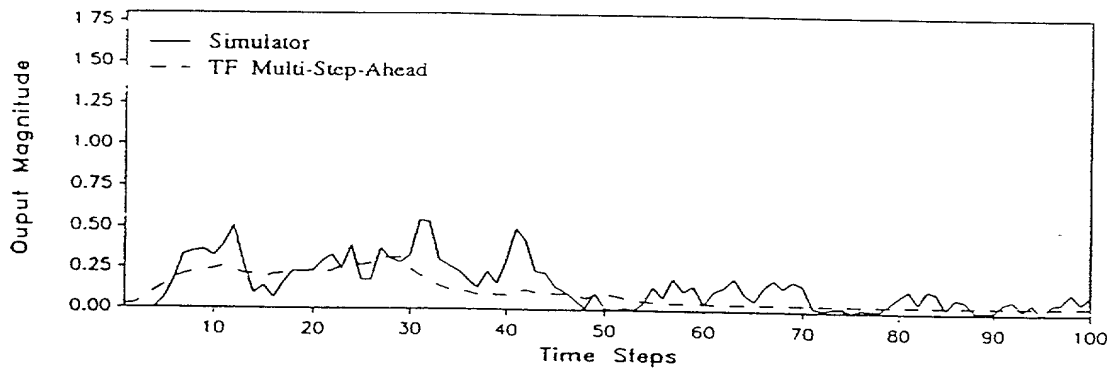
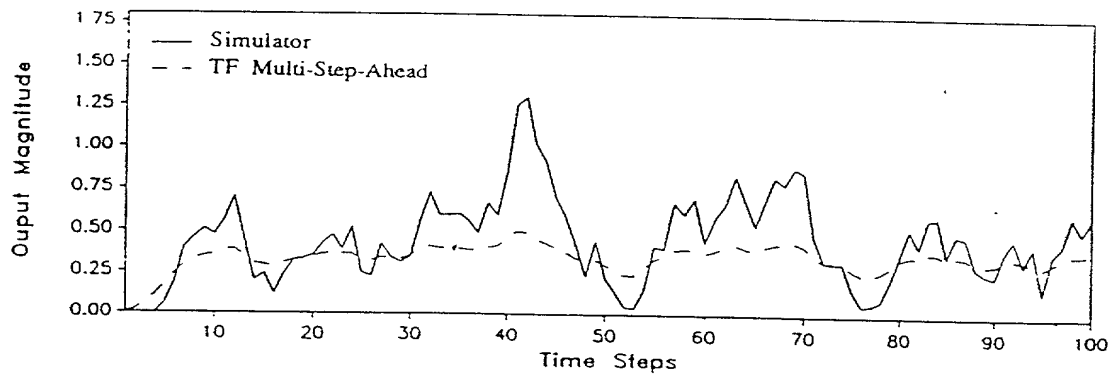
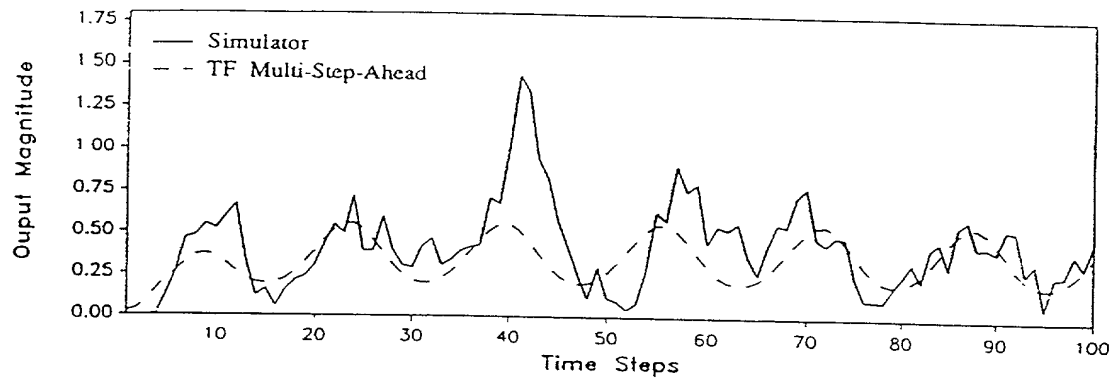


Figure 27. Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

Table 2. Relative Mean-Squared Errors for the Case-Study.

Model Structure	Sinusoidal Input	Step Input	Pulse Input
NARX SSP with $\sigma = 0.05$	0.19%	0.24%	0.17%
NARX SSP with $\sigma = 0.08$	0.30%	0.28%	0.33%
NARX MSP with $\sigma = 0.05$	0.39%	0.59%	0.36%
NARX MSP with $\sigma = 0.08$	0.60%	0.67%	0.66%
TF SSP with $\sigma = 0.05$	0.18%	0.18%	0.16%
TF SSP with $\sigma = 0.08$	0.28%	0.21%	0.29%
TF MSP with $\sigma = 0.05$	0.35%	0.47%	0.32%
TF MSP with $\sigma = 0.08$	0.60%	0.61%	0.56%

output two. Note that the errors for the TF model are slightly lower than the errors for the polynomial NARX model in both, the SSP and the MSP. The case of utilizing delayed inputs was also examined, and the results indicate that no improvement in generalization is gained by doing so.

IV.3.2.2 Batch Weight update vs. Individual Weight update

The following equation gives the weight update rule for both individual and batch update modes:

$$\Delta w_{[l',j][l,i]} = -\frac{\eta}{K} \sum_{k=1}^K \left(\frac{\partial E(k)}{\partial w_{[l',j][l,i]}} \right), \quad (100)$$

where K can be set to 1 for individual update, or N for batch update. This equation applies to all weight and bias updates in the network. The batch weight update mode means that the weight update is done after the network has gone over all the training set. While the individual weight update mode means that the weight update is done following each data point in the training set. The results presented in the previous

subsections were obtained using the individual update mode. In this subsection, the batch update mode is also examined.

The procedures used in this study are similar to the ones used in the individual weight update mode (see previous subsection). Figure 28 shows the MSE of the testing set using GF for the different number of delayed outputs utilized in the input layer of the network. Note that for each delayed outputs utilized, a node search was performed. Figure 28 presents the best performance for each case. As it can be seen from this figure, all of the networks converged to their lowest error within 10,000 iterations. The lowest MSE error obtained in Figure 28 is 0.012, for the (2-3-2) network using one delayed output, while the lowest MSE error obtained in Figure 19 is 0.0045 also for the (2-3-2) network using one delayed output. Note that none of the curves in Figure 28 went below the 0.01 mark, while all the curves in Figure 19 converged to values below the 0.01 mark. These results clearly indicate that the individual update mode gives consistently better results than the batch update mode. No satisfactory explanation has been found yet to explain these results.

IV.3.2.3 Teacher Forcing vs. Global Feedback

These two techniques were discussed in details in the previous chapter. In TF the observed value of the delayed output is utilized, while in GF the predicted value of delayed output is utilized. This latter approach is believed to perform better than the former in MSP, as it was explained earlier. In subsection IV.3.2.1 the TF with one delayed output was found to give good results. In this subsection the GF with one delayed output will be examined. The study was done using the individual weight

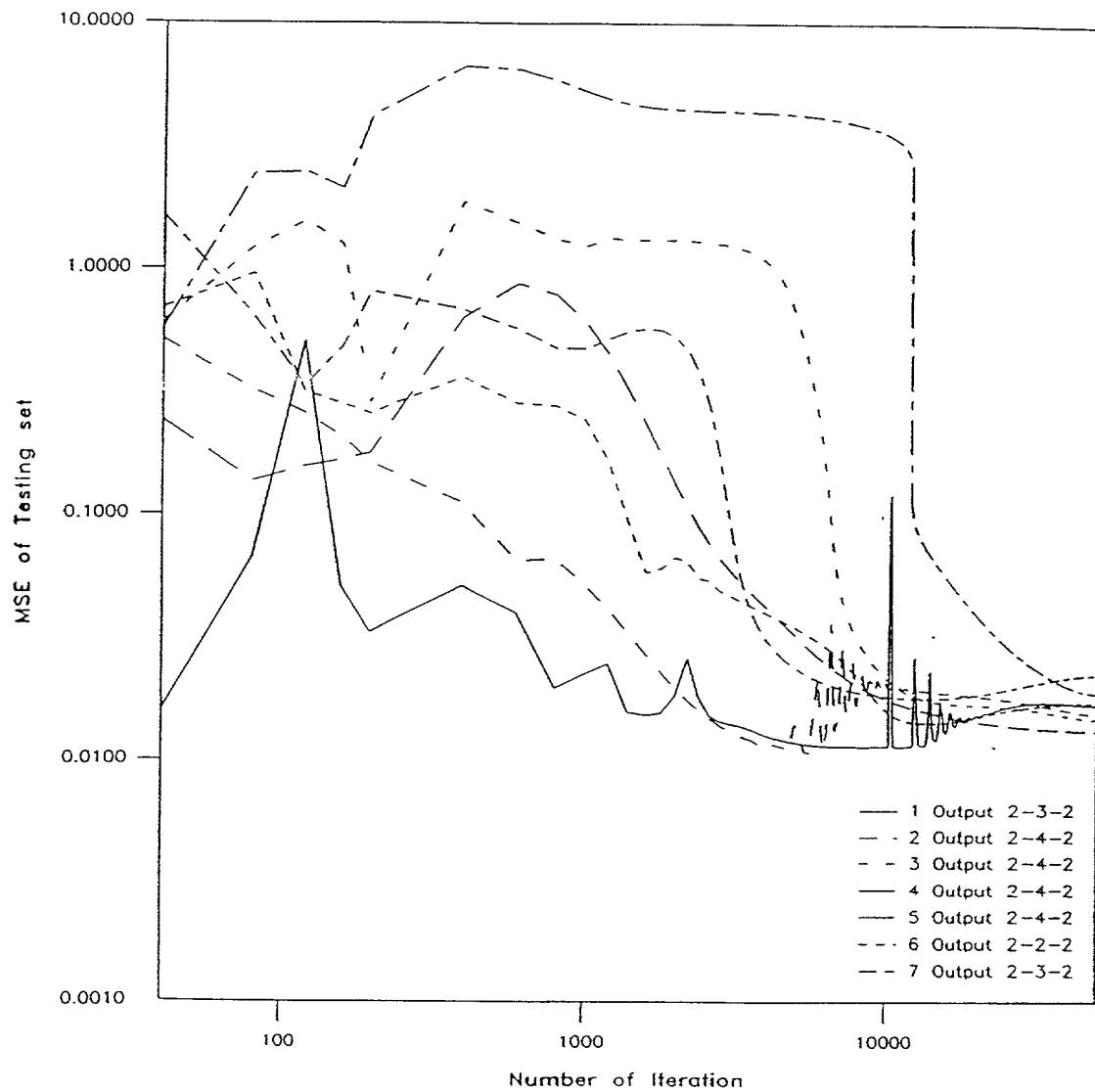


Figure 28. Mean Square Error of the Testing Set Using Global Feedback for Different Number of Delayed Outputs Utilized in the Input Layer. Batch Weight Update Mode.

update mode. The learning rate used is $\eta = 0.0005$. A node and a weight seed search were performed. The initialization of the weights and biases is a major factor for the success of the GF technique, because if the CNN is inappropriate initialized it cannot learn any of the training data set. This is because of the absence of the TF to take corrective action. The network which gave the lowest error in the testing test is the (2-3-2) network with one delayed output utilized. This network is used to generate the results presented in Figure 29 through Figure 36.

Figures 29 through 32 give the SSP of the system outputs to the three tests chosen as validation sets, while Figures 33 through 36 give the MSP of the system outputs to the same tests. The relative REMSE for each of these tests is summarized in Table 3. The errors displayed are the average errors of output one and output two. This table contains the REMSE of the polynomial NARX method, the REMSE of the TF CNN method, and the REMSE of the GF CNN method. Table 3 results in the following conclusions:

- (1) The REMSEs resulting from the GF technique are lower than the ones resulting from the TF and the polynomial NARX methods in MSP, for both low and high noise environments. This is true for all the three tests sets examined.
- (2) The REMSEs resulting from the TF method are lower than the ones resulting from the GF and the polynomial NARX methods in SSP, for low noise and high noise environments. This is also true for all the three tests shown in Table 3.
- (3) The REMSE for low noise environment are lower than REMSE for high noise environment. These results are consistent in the polynomial NARX, results as well as the CNN results (for both GF and TF).

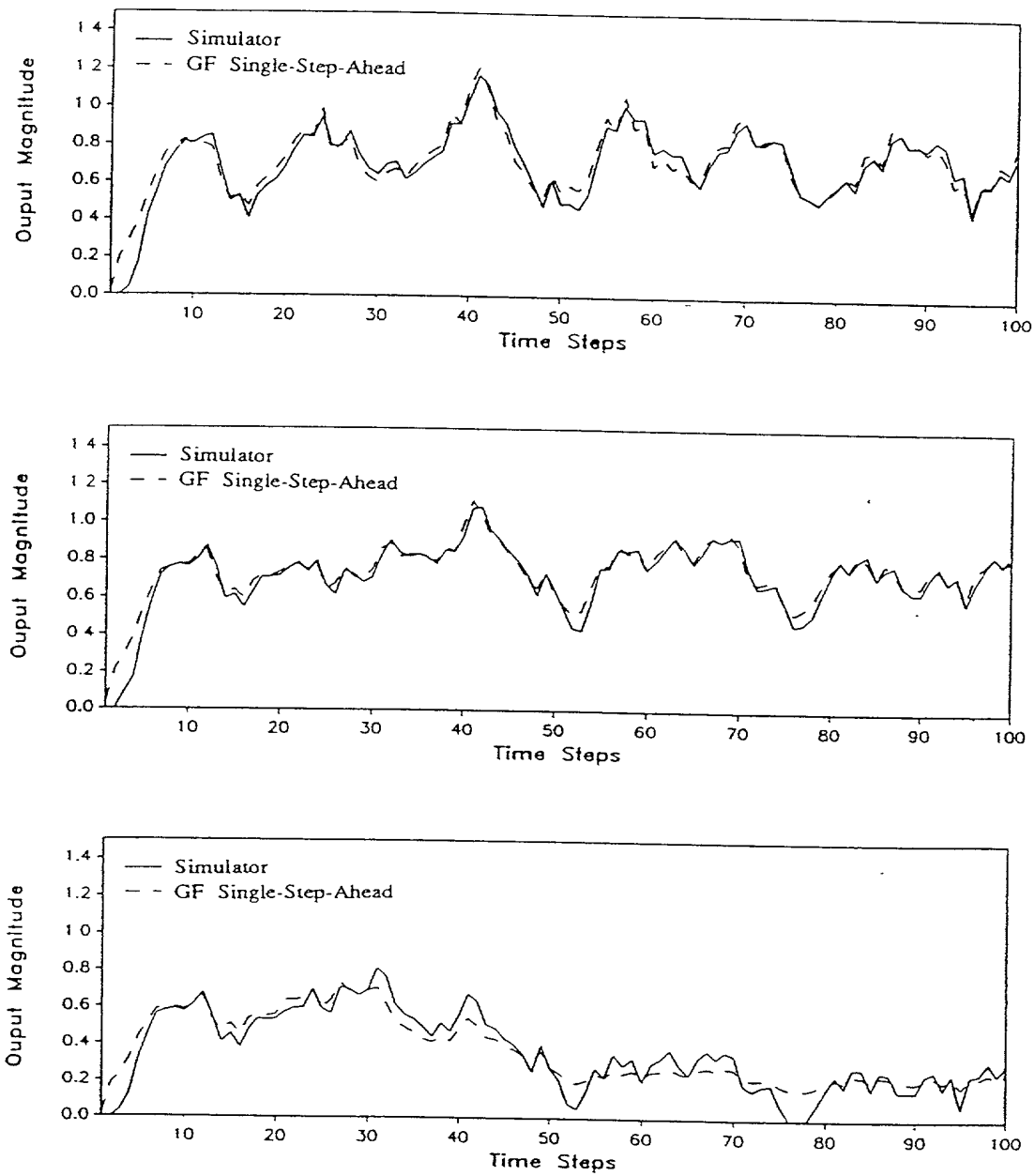


Figure 29. First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

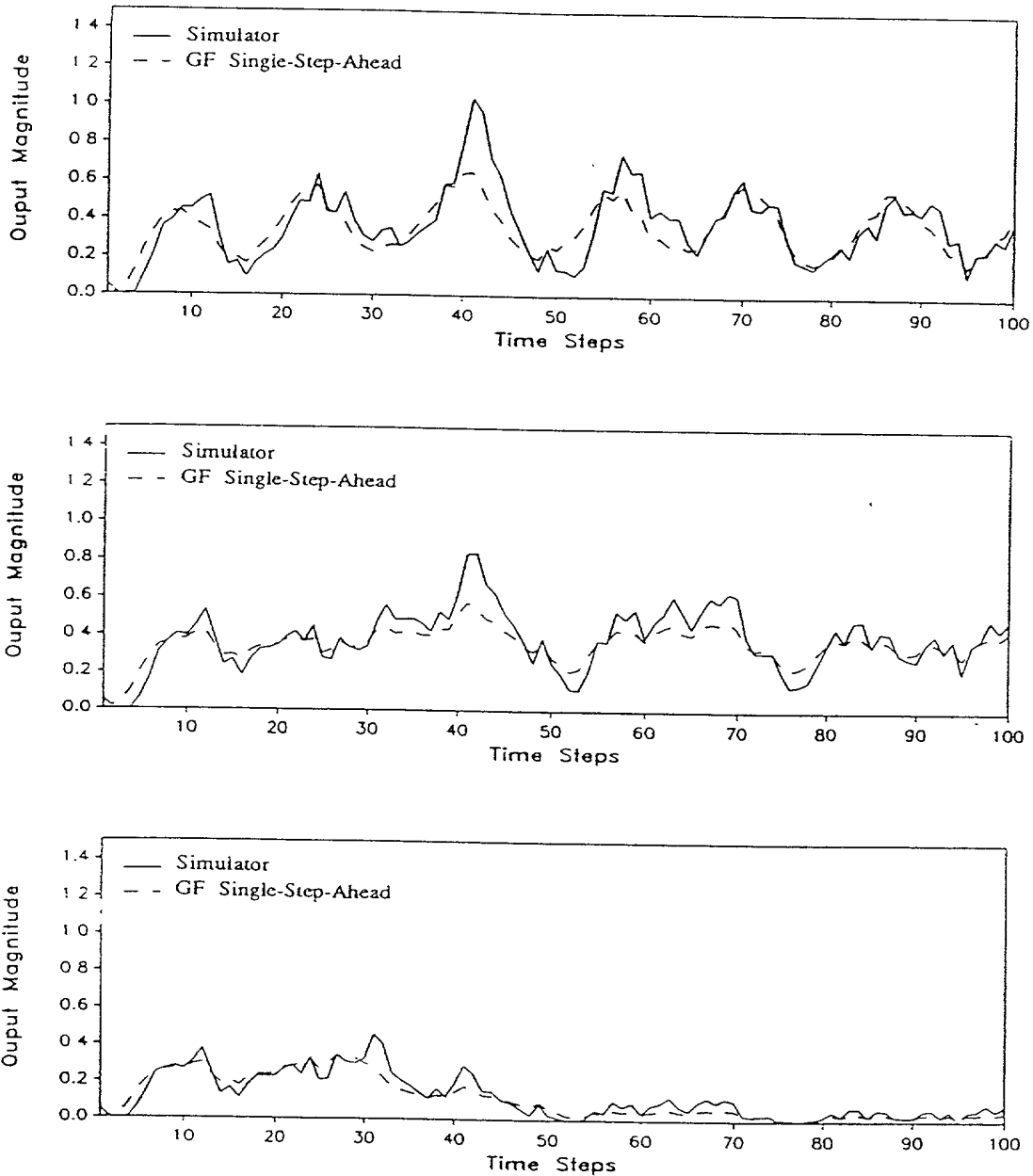


Figure 30. Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

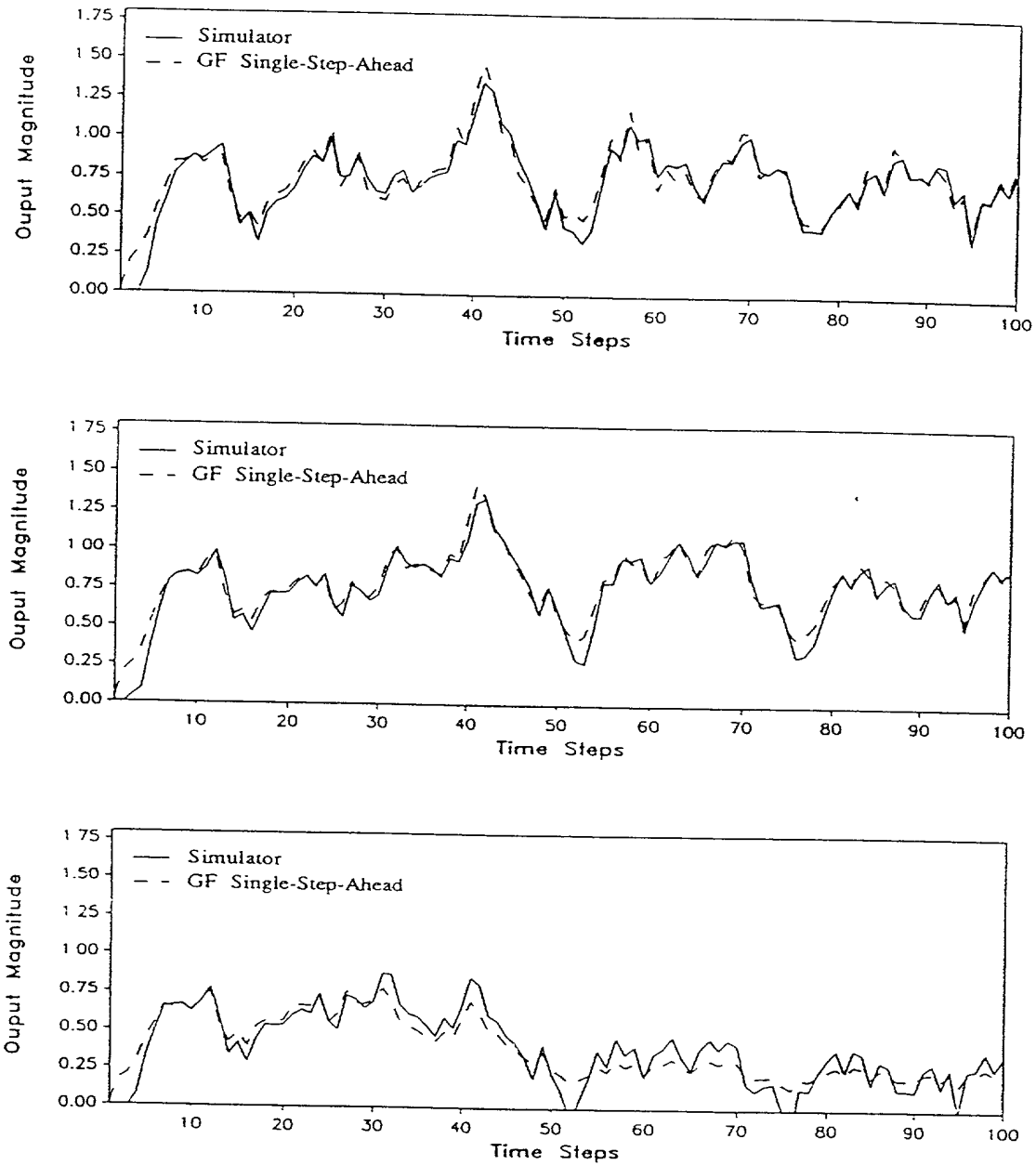


Figure 31. First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

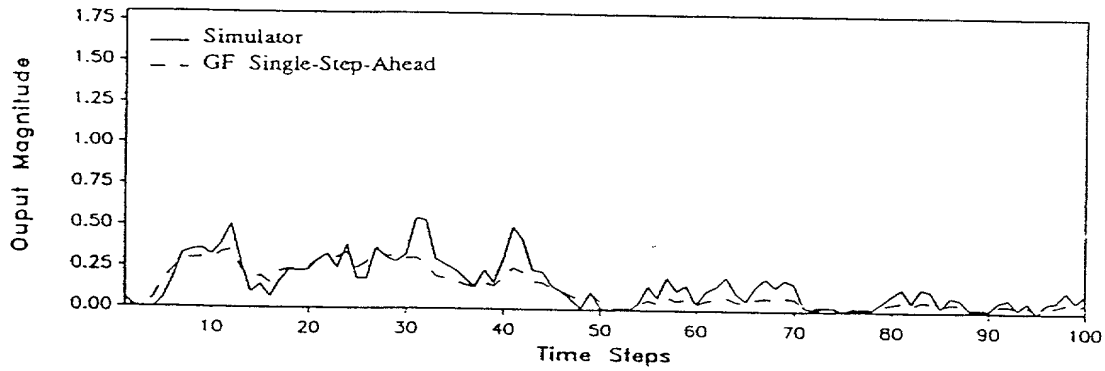
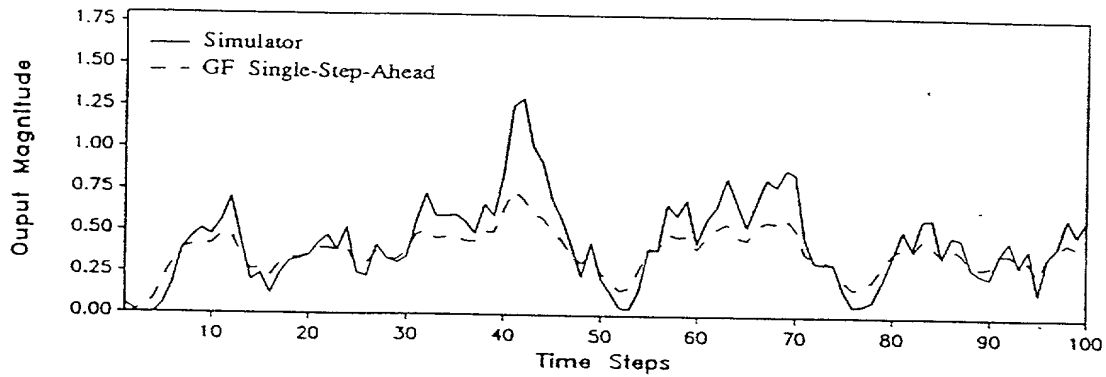
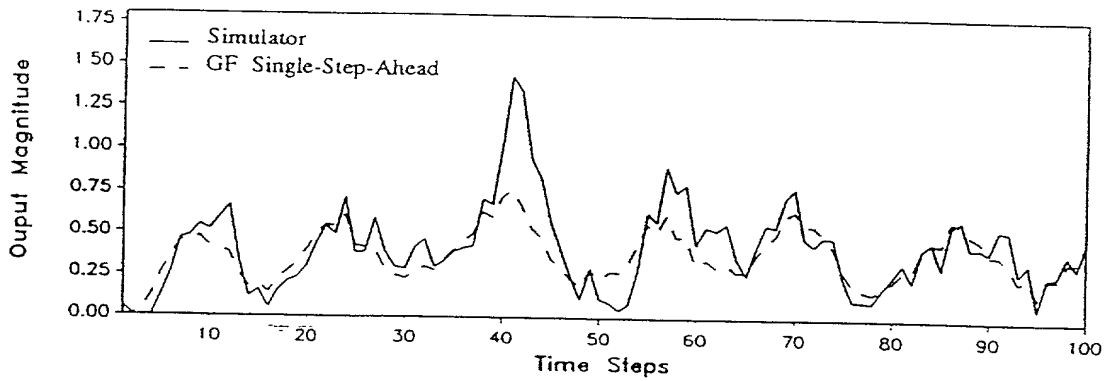


Figure 32. Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input, Middle: Step Input : Bottom: Pulse Input.)

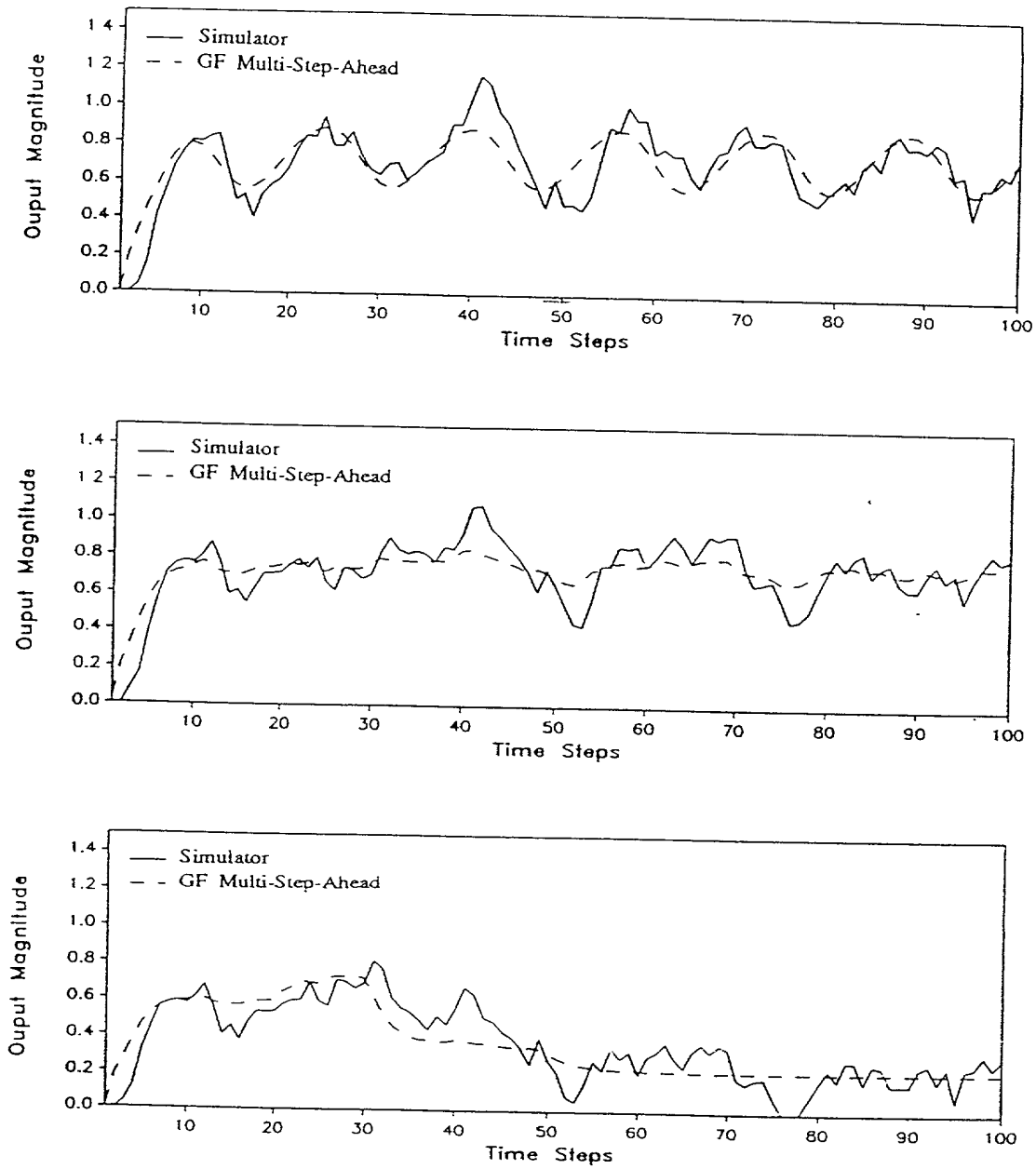


Figure 33. First System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

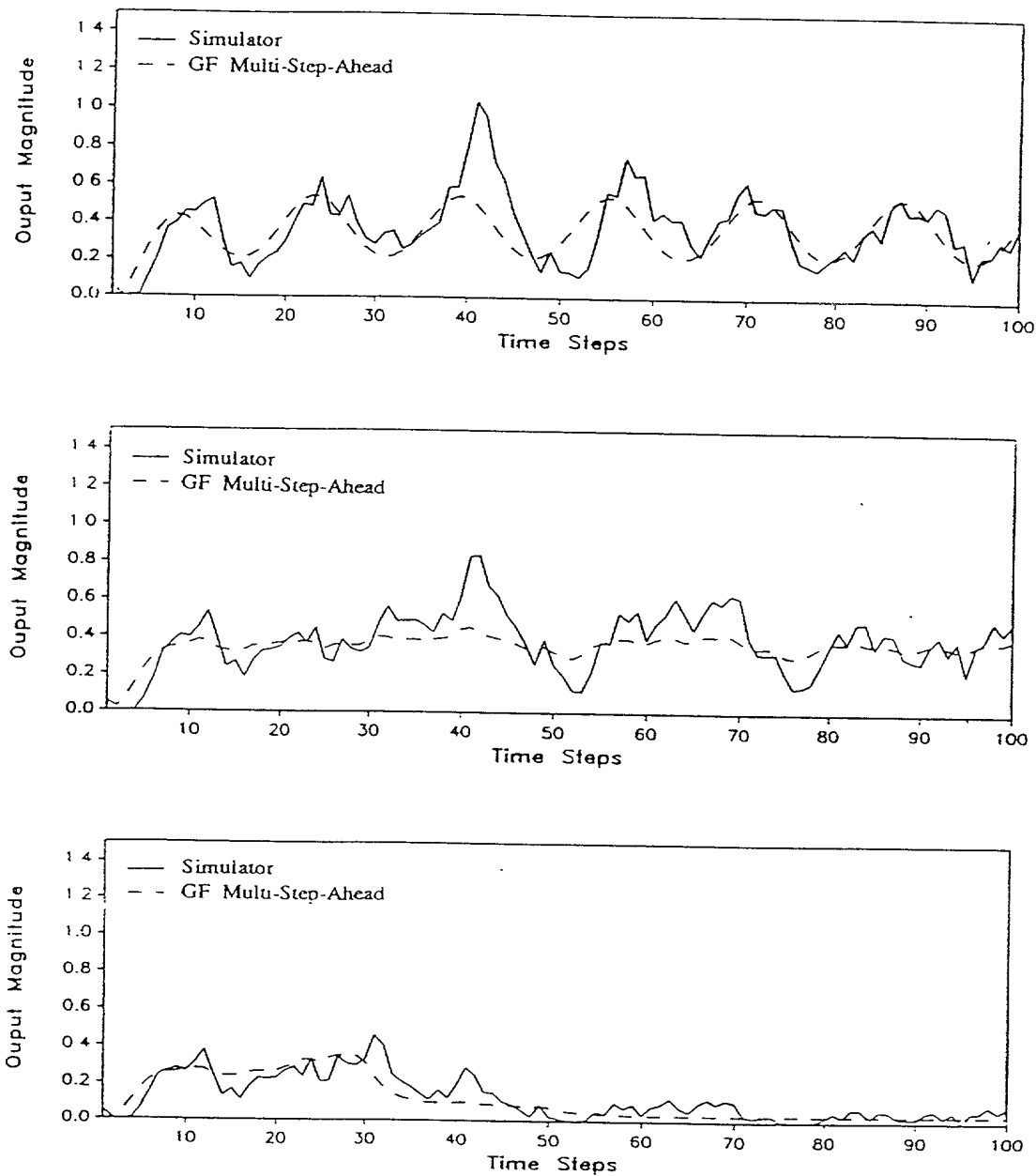


Figure 34. Second System Output Response in Low Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

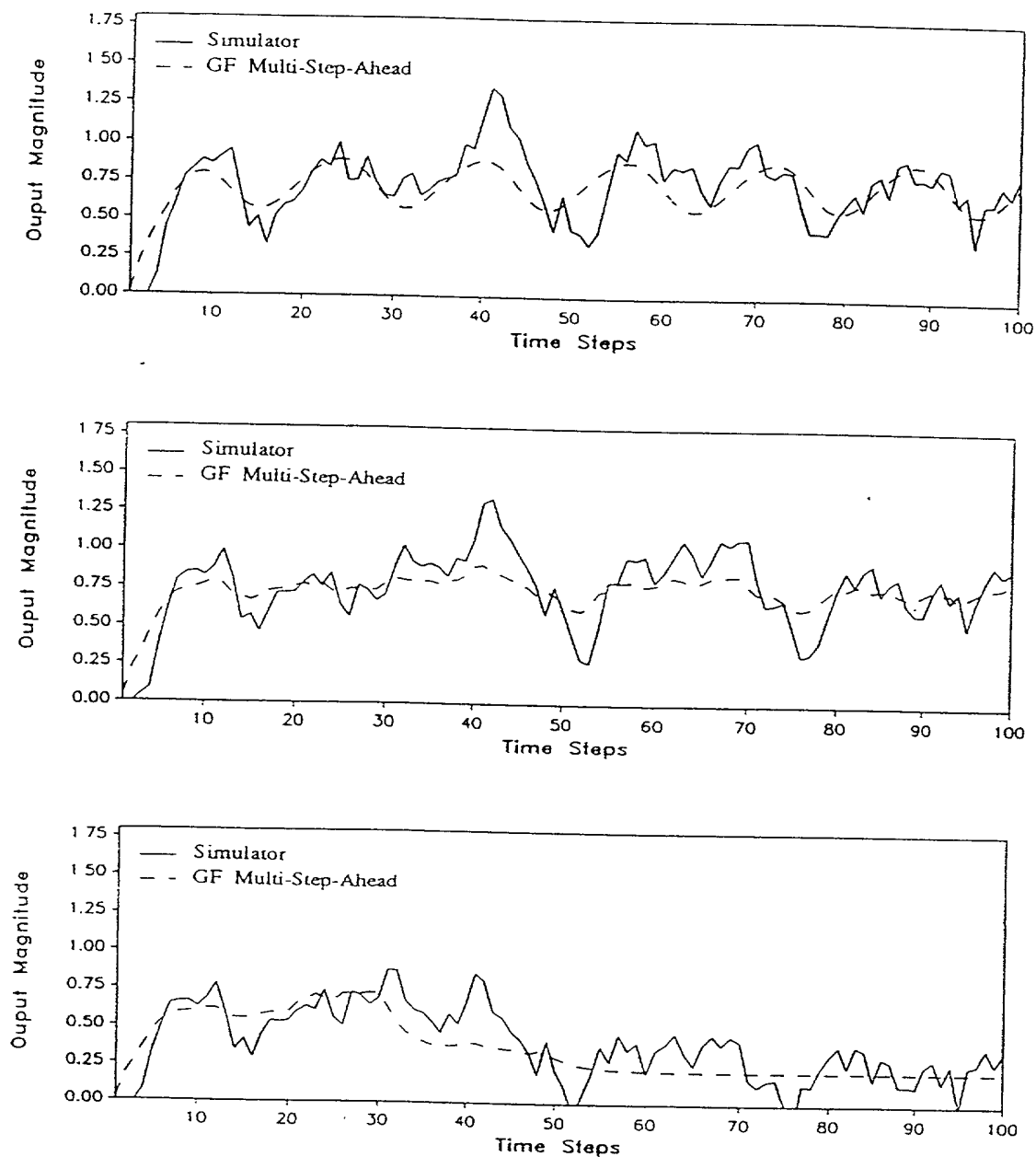


Figure 35. First System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor; (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

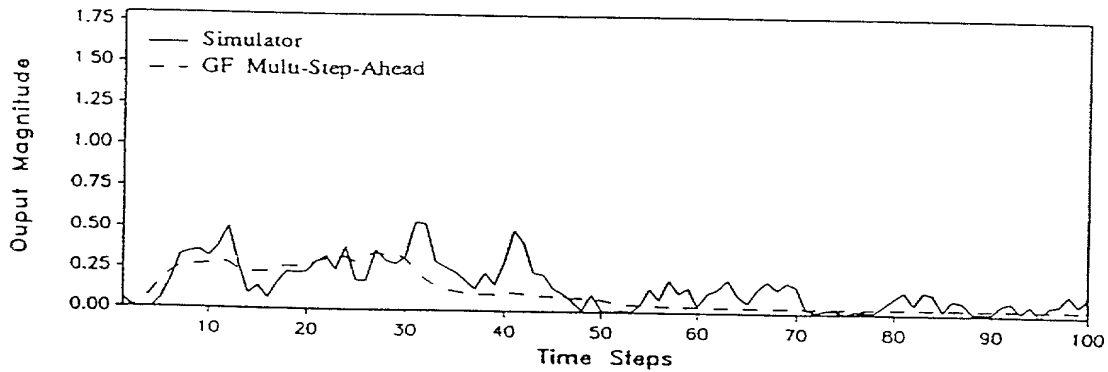
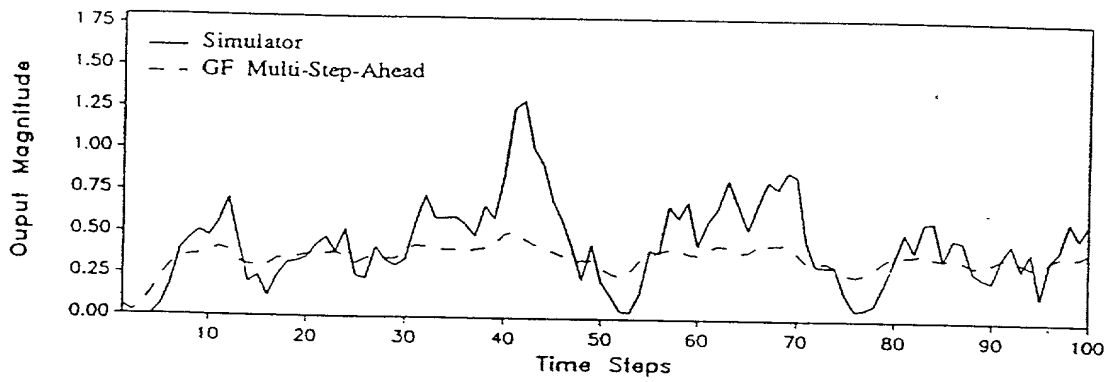
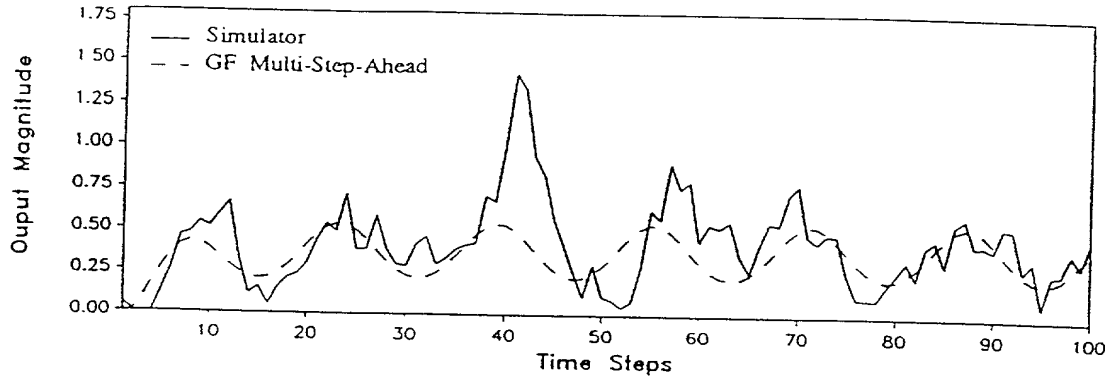


Figure 36. Second System Output Response in High Noise Environment using the Simulator and the Polynomial NARX Predictor: (Top: Sinusoidal Input. Middle: Step Input : Bottom: Pulse Input.)

Table 3. Relative Mean-Squared Errors for the Case-Study

Model Structure	Sinusoidal Input	Step Input	Pulse Input
NARX SSP with $\sigma = 0.05$	0.19%	0.24%	0.17%
NARX SSP with $\sigma = 0.08$	0.30%	0.28%	0.33%
NARX MSP with $\sigma = 0.05$	0.39%	0.59%	0.36%
NARX MSP with $\sigma = 0.08$	0.60%	0.67%	0.66%
TF SSP with $\sigma = 0.05$	0.18%	0.18%	0.16%
TF SSP with $\sigma = 0.08$	0.28%	0.21%	0.29%
TF MSP with $\sigma = 0.05$	0.35%	0.47%	0.32%
TF MSP with $\sigma = 0.08$	0.60%	0.61%	0.56%
GF SSP with $\sigma = 0.05$	0.22%	0.22%	0.17%
GF SSP with $\sigma = 0.08$	0.36%	0.28%	0.32%
GF MSP with $\sigma = 0.05$	0.34%	0.41%	0.28%
GF MSP with $\sigma = 0.08$	0.58%	0.55%	0.54%

- (4) The REMSE for the SSP are much lower than REMSE for the MSP. These results are also consistent throughout the polynomial NARX method, as well as the CNN methods.

The reason the errors of TF for SSP are lower than the errors for GF in SSP is of the following: In TF the emphasis of the training, i.e. calculation of the gradients, is the SSP, although the weights are determined based on the network performance in MSP; In GF, on the other hand, the emphasis of the training is MSP, i.e. the gradients are calculated to account for the MSP effects. This also explains the improved transient performance of the GF compared to the TF in MSP.

IV.3.2.4 Weight Decay During Training

Weight decay is the technique used in removing redundant connections during training to prevent overtraining. This can be achieved by giving each connection

weight w_{ij} a tendency to decay to zero. This method allows the network to remove some of its connections, and as a result it reduces the complexity of the network. This is achieved by the following equation [24]:

$$w_{ij}^{new} = (1 - \lambda)w_{ij}^{old}, \quad (101)$$

where λ is some small parameter. This is equivalent to adding a penalty (or regularization) term to the original cost, or objective function E_o , in equation (35) as follows:

$$E \equiv E_o + \frac{1}{2}\gamma \sum_{ij} \frac{w_{ij}^2}{1 + w_{ij}^2}, \quad (102)$$

where the gradient descent is performed on the total error E as follows :

$$\Delta w_{ij} = -\eta \partial E / \partial w_{ij}.$$

The parameter λ which depend on w_{ij} is then given by:

$$\xi_{ij} = \frac{\gamma\eta}{(1 + w_{ij}^2)^2}, \quad (103)$$

where η is the learning rate, and γ the weight decay coefficient. This decay rule was used in the 2I2O case study. The γ , which is a variable parameter, was varied from 0.001 to 0.05, and different CNN structures were chosen to investigate this option. The individual weight update mode was chosen over the batch update mode, because it was previously decided that the former outperforms the latter. Furthermore, only one delayed output was utilized, because there was no benefit in feeding more delays to the CNN input layer. Note that only the TF option was used for investigating the weight decay technique.

The values of the weights did not decay as it was expected, and the performance of the network with $\gamma = 0$ was always better than the performance of the same network with any value larger than 0. These results are consistent with reports in the literature that weight decay has been shown to improve generalization in feedforward networks but not in recurrent networks [29]. Finally, it is worth mentioning that there have been studies indicating that adding a regularization term in the objective function, i.e. weight decay, has equivalent impact on generalization as does cross-validation, i.e. out-of-sample testing [58]. The latter is the approach used in this research study, and it clearly appears to be more effective than regularization.

IV.3.2.5 Polynomial NARX Model Structure Use in CNNs

In section IV.2 of this chapter the polynomial NARX model was investigated, and its results were presented. In this section, the model structure obtained using the polynomial NARX method will be used as input the CNN. The polynomial NARX model structure found is:

$$\begin{aligned} \hat{y}_1(k/k-1) = & 0.79y_1(k-1) + 0.3u_1(k-1) - .43y_1(k-3)u_1(k-2) \\ & + 1.3u_1^2(k-2)u_2(k-1) + 0.17y_1(k-3), \end{aligned} \quad (104)$$

$$\begin{aligned} \hat{y}_2(k/k-1) = & 0.69y_2(k-1) + 0.44y_1(k-1)u_1(k-1) + 0.04y_1(k-4) \\ & - 1.08u_1^2(k-3)u_2(k-3) - 0.66u_1(k-4)u_1(k-2)y_2(k-2). \end{aligned} \quad (105)$$

The logic behind this idea is simple. Since the polynomial NARX method detects the terms which are most relevant to the system, these terms should be the appropriate terms to utilize in the input layer of the CNN. Figure 37 shows a CNN, for which the input layer is made-up of terms given in equations (104) and (105). Given this set-up, a node search was performed to find the number of nodes which give the lowest error in the testing data set. In this investigation only the TF option was investigated, however, both the individual and the batch weight update modes were examined. For the individual weight update mode the (2-4-2) network is the one which gives the lowest error in the testing data set. For the batch weight update mode, the (2-2-2) network is the best one. Figure 38 gives the MSE of these two networks, as well as the best results obtained using the previously described approaches. As can it be seen from this figure, neither the (2-2-2) network nor the (2-4-2) gave any better results than the results obtained using the TF with one delayed output utilized.

IV.4 Chapter Summary

In the case-study of a 2I2O system presented in this section, both the most promising conventional method and the CNN methods of SI were examined and compared. The objective was to build a model capable of accurate MSP. The conventional method examined was the polynomial NARX method which is believed to be the most powerful method in conventional nonlinear SI. In CNN methods many options were considered and examined. Individual weight update was found to perform better than batch weight update. GF was found to outperform TF in MSP, but not in

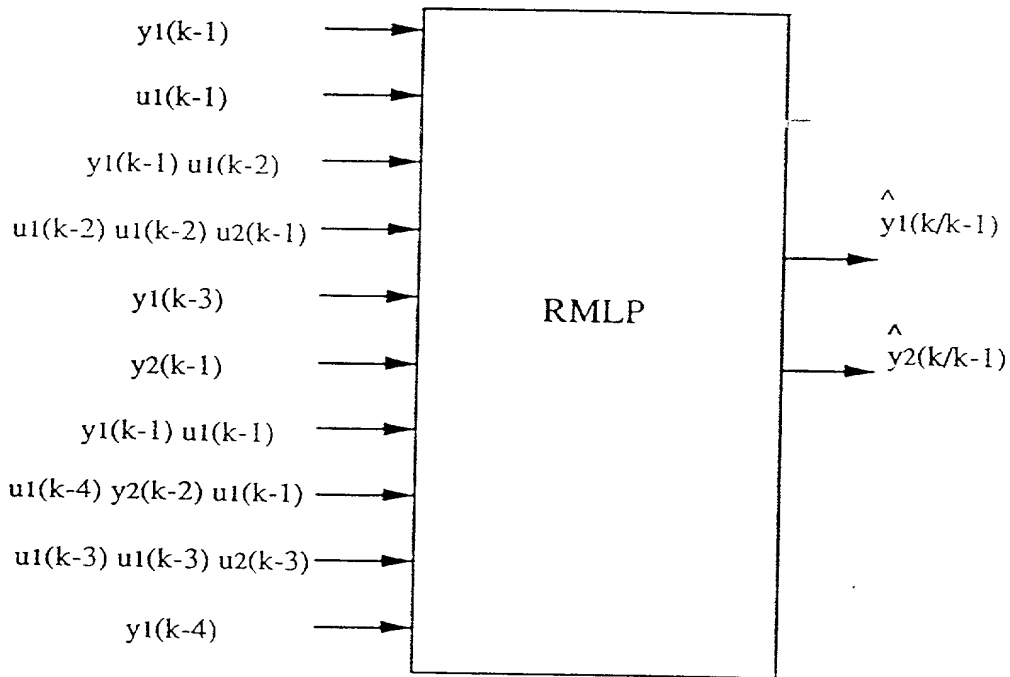


Figure 37. A CNN which the Input Layer Consisting of Terms Obtained from the Polynomial NARX Model Structure Detection Algorithm.

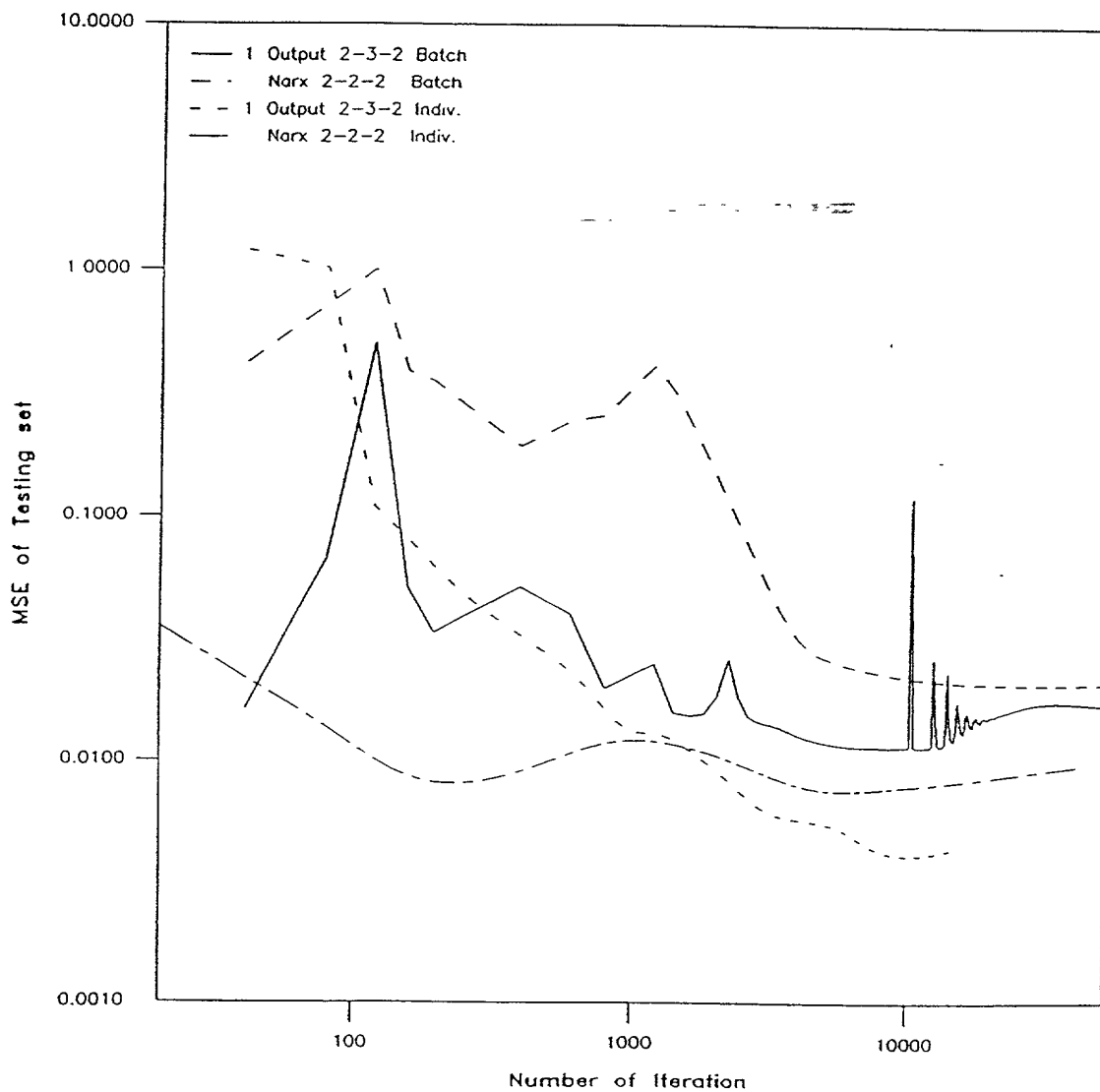


Figure 38. Mean Square Error of the Best Networks, Utilizing CNN With Teacher Forcing.

SSP. Weight decay, was also examined but did not yield acceptable results. Finally, incorporation of the polynomial NARX structure detection method with CNNs did not improve the network's ability to perform accurate MSP.

From these comparative results, it is clearly demonstrated that CNN methods outperform the best of the conventional SI method consistently. However, the improvement gained by using CNNs was not very significant. because in this case study the 2I2O system is not a complex system, rather an artificial test case. The impact of the CNN methods in SI will be demonstrated by developing a predictive model capable of MSP for the UTSG, a highly complex process system. This will be the objective of the next chapter.

U-TUBE STEAM GENERATOR EMPIRICAL MODELING

V.1 Introduction

In this chapter, the nonlinear empirical modeling of a U-Tube Steam Generator (UTSG) encountered in a nuclear power plant is addressed, in an input-output sense, using a conventional System Identification (SI) technique, the polynomial NARX, and CNN methods. The objective is to design a CNN-based model of the UTSG capable of performing accurate multi-step-ahead prediction (MSP) as well as single-step-ahead prediction (SSP).

The polynomial NARX method was presented in details in section IV.2, and it was applied successfully to the case study in chapter IV. The developed CNN methods were presented in chapter III, and further investigated in chapter IV. Two learning methods were investigated, namely, the Teacher Forcing (TF) method, and the Global Feedback (GF) method. The learning algorithms were presented and discussed in details in chapter III. Also both algorithms were successfully applied to the case study presented in chapter IV.

The remainder of this chapter is organized as follows: A brief description of the UTSG is described in the next section. Section VI.3 presents a description of the UTSG simulator adopted for the purpose of this research study. Section VI.4 presents the empirical modeling of the UTSG using the polynomial NARX method, the TF CNN method, and the GF CNN technique. Section VI.5 is the chapter summary.

V.2 Description of the U-Tube Steam Generator

A schematic diagram of a Westinghouse type UTSG is shown in Figure 39. High pressure liquid from the plant primary-loop flows in and out of the steam generator primary side (tube side). This is indicated in the figure as *Hot Leg in* and *Cold Leg out*. The primary side liquid flow path is up from the hot leg inlet, to a semicircular turnaround, and then down to the cold leg exit.

—Feedwater enters the UTSG from the steam plant, as indicated by the arrow *Feed-water In*, through a normally submerged feed ring. At this junction, the feedwater mixes with the liquid being discharged from the liquid-vapor separation devices. The liquid mixture flows downward through the annular *Downcomer* region. After a turn at the bottom of the Downcomer, the liquid is heated during an upward passage outside the U-Tubes in the *Tube Bundle region*. The heat transferred across the tube bundles causes evaporation of some of the liquid, and a two-phase mixture exits from the tube bundle entering the *Riser*. The liquid and vapor are partially separated by swirl vanes at the top of the riser and more separation occurs near the top of the UTSG. The vapor that results from this is indicated by *Steam Out*, and it is saturated with a quality of about 99.75%. This vapor is then fed to the turbine units and other auxiliary equipments [62].

V.3 Description of the Available UTSG Simulator

An existing UTSG simulator developed by Strohmayer [60], and modified by Choi [15] was adopted for the purpose of this study. The simulator was developed using

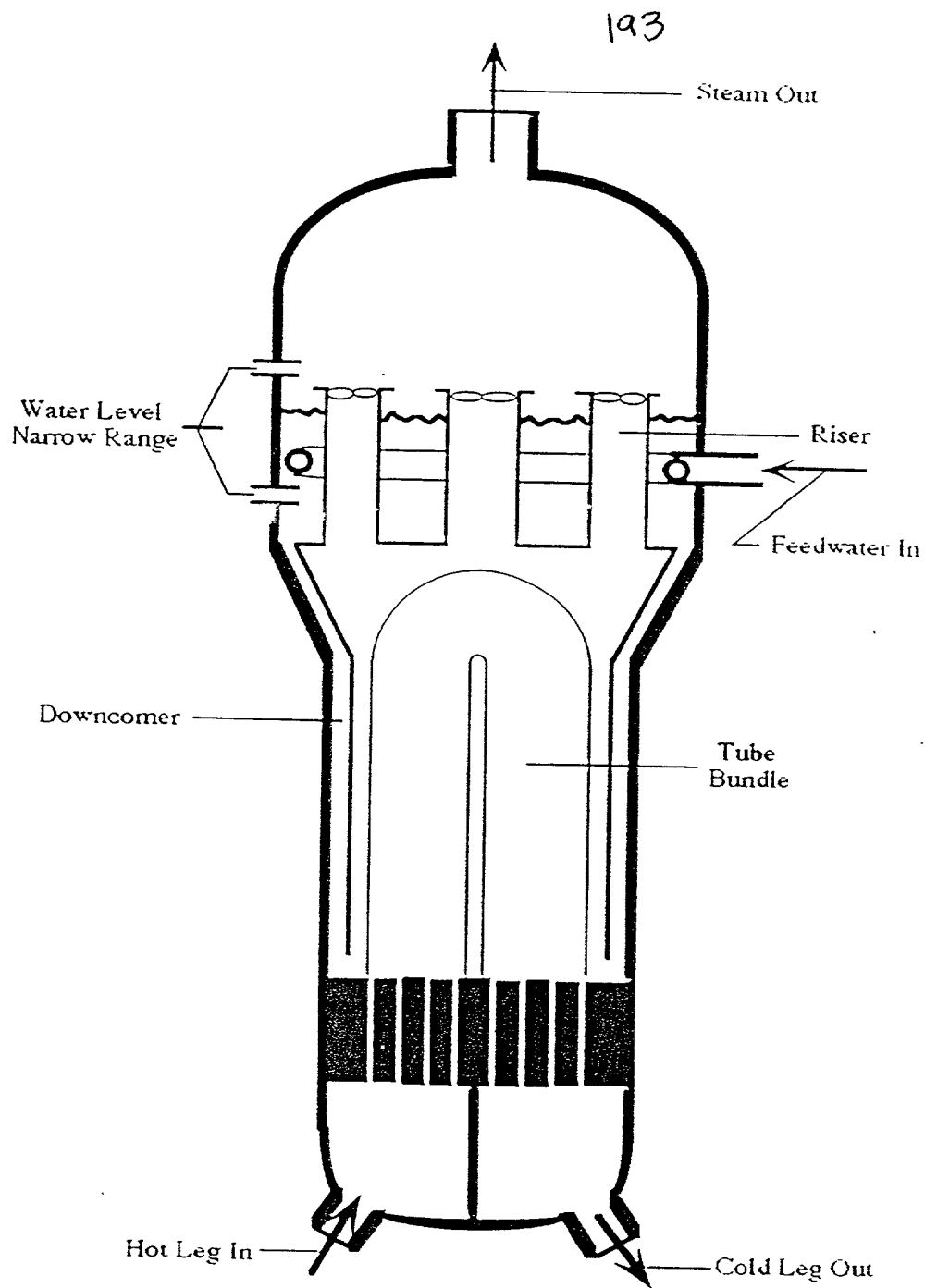


Figure 39. Schematic Diagram of a U-Tube Steam Generator.

one-dimensional mass, momentum, and energy conservation equations. An integrated secondary-recirculation-loop momentum equation is incorporated into the simulator to calculate the water level. Because the open-loop system is unstable, a stabilizing controller is required to allow system operation. The controller structure used with the above UTSG is the one proposed by Menon and Parlos [62].

The UTSG has one control input which is the feedwater flow rate W_{fw} , and five disturbance inputs namely, the hot-leg temperature T_{hl} , the primary flow rate W_{pr} , the steam flow rate W_{st} , the feedwater temperature T_{fw} , and the primary pressure P_{pr} . The measured outputs of the UTSG are three, namely, the water level L_w , the secondary pressure P_{sat} , and the cold leg temperature T_{cl} . Figure 40 shows a block diagram of the UTSG with the input, the disturbances, and the outputs denoted.

The adopted UTSG simulator has three control volumes (regions) on the primary side and four control volumes on the secondary side. The primary side region consists of the inlet plenum, the fluid volume within the tubes of the tube bundle, and the outlet plenum. The four secondary side regions are: the tube bundle region; the riser region; and the steam dome-downcomer, which is divided into a saturated volume and a subcooled volume. The saturated and subcooled volumes have a movable interface, the position of which is an unknown variable.

This complex process system is made-up of nine nonlinear ordinary differential equations. The six states of the secondary-loop of the UTSG are: the internal energy at the downcomer exit, the vapor volume in the steam dome, the void fraction at the riser inlet and outlet, the steam pressure, and the recirculation flow rate, respectively. While the other three states of the primary-loop of the UTSG are: the inlet

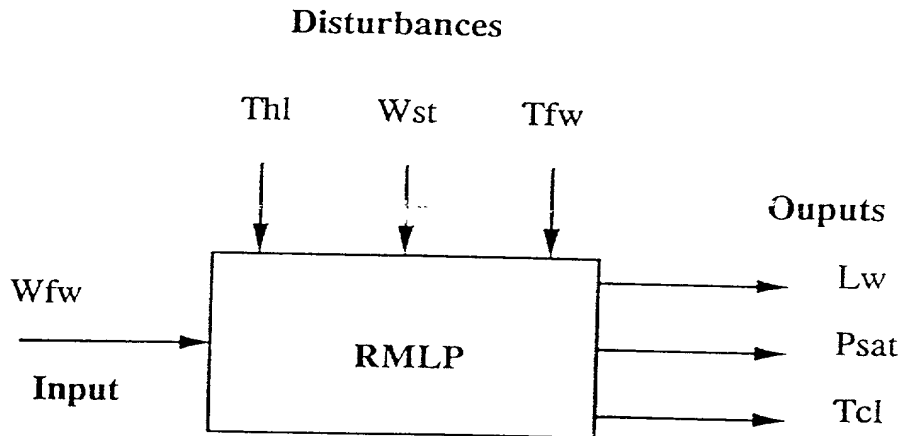


Figure 40. Block Diagram of the UTSG with Input, Disturbances, and Measured Outputs.

temperature at the tube inlet, the outlet temperature at the tube outlet, and the outlet temperature at the cold leg.

The water level of a UTSG is always maintained between preset limits to assure proper operation. If the UTSG water level is not maintained within such preset limits, a reactor scram is initiated by the plant protection system. The water level control problem is complicated by the nonminimum phase effects known as “shrink” and

“swell”, which occur during plant transients and which are more prominent at start-up/low power operation. Because of the presence of steam bubbles in the tube bundle region of a UTSG, the water level measured in the downcomer region temporarily reacts in a reverse manner to water inventory changes. In addition to the high process and sensor noise associated with the operation of the UTSG, reverse dynamics make this a highly complex process system.

V.4 U-Tube Steam Generator Modeling

Based on the results obtained from the case study in the previous chapter, it is clear that CNN has the potential of identifying nonlinear systems. However, the polynomial NARX method of conventional SI theory is also investigated. The objective is to compare the conventional method (polynomial NARX) and the CNN methods (TF and GF) for this highly complex process system.

As stated previously, a UTSG simulator is used to generate the necessary data needed to develop an empirical model. This simulator has been successfully validated against plant data for the entire power range of operation of the UTSG [15]. The empirical CNN model developed in this research study consists of six separate sub-models. Three models to predict the water level response at low power, medium power, and high power operation, respectively, and three other models to predict the secondary pressure response at low power, medium power, and high power operation, respectively. The low power range is from 0% to 20%, the medium power range is from 20% to 50%, while the high power range is from 50% to 100% of full power operation.

The reason for breaking-up the operating range of the UTSG is the varying degree of complexity for each power range. For example the nonminimum phase effects are much more pronounced at the low power levels than at any other power levels. Moreover, having three separate power ranges increases the performance of each model within its own range. Another reason for having three separate power ranges is to accommodate for the data magnitude within each range. The magnitudes of the data at high power levels are much larger than the ones at the low power levels. As all the data are scaled by the same factors, the low power range data will have numbers in the order 0.001 while the high power range data will have numbers in the order of 0.1.

A combination of steps and ramps were generated at each power range to be used as training and testing data set. The sampling interval for all the data set collected is 5 sec. The following steps were generated for the low power range: 5% to 10%, 10% to 15%, and 15% to 20% of full power. Similarly the following ramps were generated for the low power range: 5% to 20%, and 20% to 5% of full power, with a magnitude rate of 2% of full power per minute. The testing data set consists of a ramp decrease from 15% to 5% of full power at the rate of 2% of full power per minute. The total number of samples for the low power range is 2500. The following steps ramps were generated for the medium power range: 20% to 50% of full power with step power increments of 5%. Similarly, the following ramps were generated for the medium power range: 20% to 50%, and 50% to 20% of full power with a magnitude rate of 6% of full power per minute. The testing data set consists of a ramp decrease from 45% to 35% of full power at the rate of 6% of full power per minute. The total number of samples for

the medium power range is 2530. The following steps ramps were generated for the high power range: 50% to 100% of full power with step power increments of 5% of full power. Similarly, the following ramps were generated for the high power range: 50% to 100%, and 100% to 50% of full power with a rate magnitude of 8% of full power per minute. The testing data set consists of a ramp decrease from 85% to 65% of full power at a rate magnitude of 8% of full power per minute. The total number of samples for the high power range is 1700. All these data sets were generated with high process and sensor noise which was incorporated into the UTSG simulator.

Following data collection scaling was performed for all the data generated. The scaling part is a major factor in designing CNN models. This is because the squashing function, a hyperbolic tangent in this case, has an output range limited within $[-1, 1]$. The output of each hidden nodes goes through a squashing function. For large node outputs the squashing function will always be 1 or -1, thus preventing the node from learning the training data set. This effect is known as *saturation*, and it is usually prevented by performing appropriate scaling. An unscaled data set may also cause numerical problems when using conventional SI method. Scaling was performed in this research study for all four inputs and two outputs of the UTSG data set. The scaled data were generated according to following equation:

$$X_{scaled} = \frac{(X_{unscaled} - \bar{X}_{unscaled})}{X_{max_{unscaled}}}, \quad (106)$$

where $\bar{X}_{unscaled}$ is the average value of $X_{unscaled}$ over N data points, and $X_{max_{unscaled}}$ is the maximum value of $X_{unscaled}$. Note that each of the developed UTSG model was scaled separately.

The most important aspect of empirical modeling is the procedures followed for model validation. Unfortunately, there are no universal procedures for general empirical model validation, and a number of ad-hoc, problem dependent approaches are frequently used in addressing the issues associated with it. Things become even worse when attempting to validate nonlinear empirical models. Failure to satisfy model validation tests means that the empirical modeling procedure will have to be reiterated. There are a number of empirical model validation approaches available, depending upon the empirical structure selected at the model development stage. Certainly, as the complexity of the model structure increases, the available choices are significantly limited. For nonlinear empirical model structures, either input-output or state-space, model validation is usually performed via the analysis of the residuals (modeling error terms) using a variety of techniques, such as high-order correlation function analysis [6], [8], [9], [31], or via simulations using new test data, if available [34],[41], [44].

In this study only simulation using new data set are used for model validation. The reason behind rejecting the high-order correlation tests is that they are not applicable to models designed for MSP. Billing's correlation test checks the whiteness of the residuals, provided that the process noise associated the system under study is also modeled. To model the process noise, the residuals at each time step must be available as input to the network. However, the residuals are not available while performing MSP.

Therefore, in this study, extensive simulation tests were performed for validating the UTSG empirical models. New data set at various ranges of normal power operation are collected. Several steam flow rate steps with different magnitudes and various

steam flow rate ramps with different slopes are applied to the UTSG in order to check the validity of the proposed empirical CNN models. The following section gives the results obtained using the polynomial NARX method and the CNN methods. Throughout this chapter the error criterion adopted is the normalized root-mean-squared error (RMSE) given by the following equation:

$$\text{Error}(i) \equiv \frac{\sqrt{\sum_{k=1}^N (\hat{y}_i(k/k-1) - y_i(k))^2}}{\sqrt{\sum_{k=1}^N y_i^2(k)}}; i = 1, 2. \quad (107)$$

V.5 Computer Simulation Results and Model Validation

This section presents the results obtained using the polynomial NARX method, the TF CNN method, and the GF CNN method. The techniques used hereby have been described in previous sections in detail. Note that the emphasis will be on the GF method since both the polynomial NARX method and the TF method failed to give an adequate model capable of performing accurate MSP.

V.5.1 Polynomial NARX Model

The polynomial NARX feed forward regression method was used to model the medium power level range for the water level response of the UTSG. Different models were obtained using the polynomial NARX method, and the model which performed the best is given by the following equation:

$$\hat{y}(k/k-1) = 0.83y(k-1) - 0.46y^2(k-1)u_1(k-1) + 1.55y^3(k-1) \quad (108)$$

y stands for the downcomer water level, while u_1 is the feedwater flow rate. This solution shows that the only input, among disturbances and inputs, relevant to the UTSG is the feedwater flow rate. This is partially true, because both the steam flow rate and the feedwater temperature have major influence on the operation of the UTSG. The failure of the polynomial NARX method in selecting the other inputs is probably due to the highly complex nature of the UTSG system, as well as because of the high level of process and sensor noise present in the data.

Figure 41 and 42 give the water level response of the polynomial NARX predictor to a step power level increase from 20% to 25% of full power, and a ramp decrease from 35% of full power level to 20% of full power level, respectively, for the cases of SSP and MSP. A step transient was used in designing the polynomial NARX model, while a ramp transient is the validation data set. Figure 41 is an indication that the water level response is somewhat satisfactory for the medium power level range for the SSP. The water level response using SSP in the transient shown in Figure 42 is also satisfactory. However, for the MSP the polynomial NARX failed to predict the water level response to the step and the ramp power level increase and decrease as shown in Figures 41 and 42. The failure of the polynomial NARX in MSP was consistent for all attempted transients. As a result, the performance of the polynomial NARX technique is definitely not satisfactory in fulfilling the MSP task, the objective of this research study. Note that no attempt has been made to model the low or high power ranges of UTSG operation.

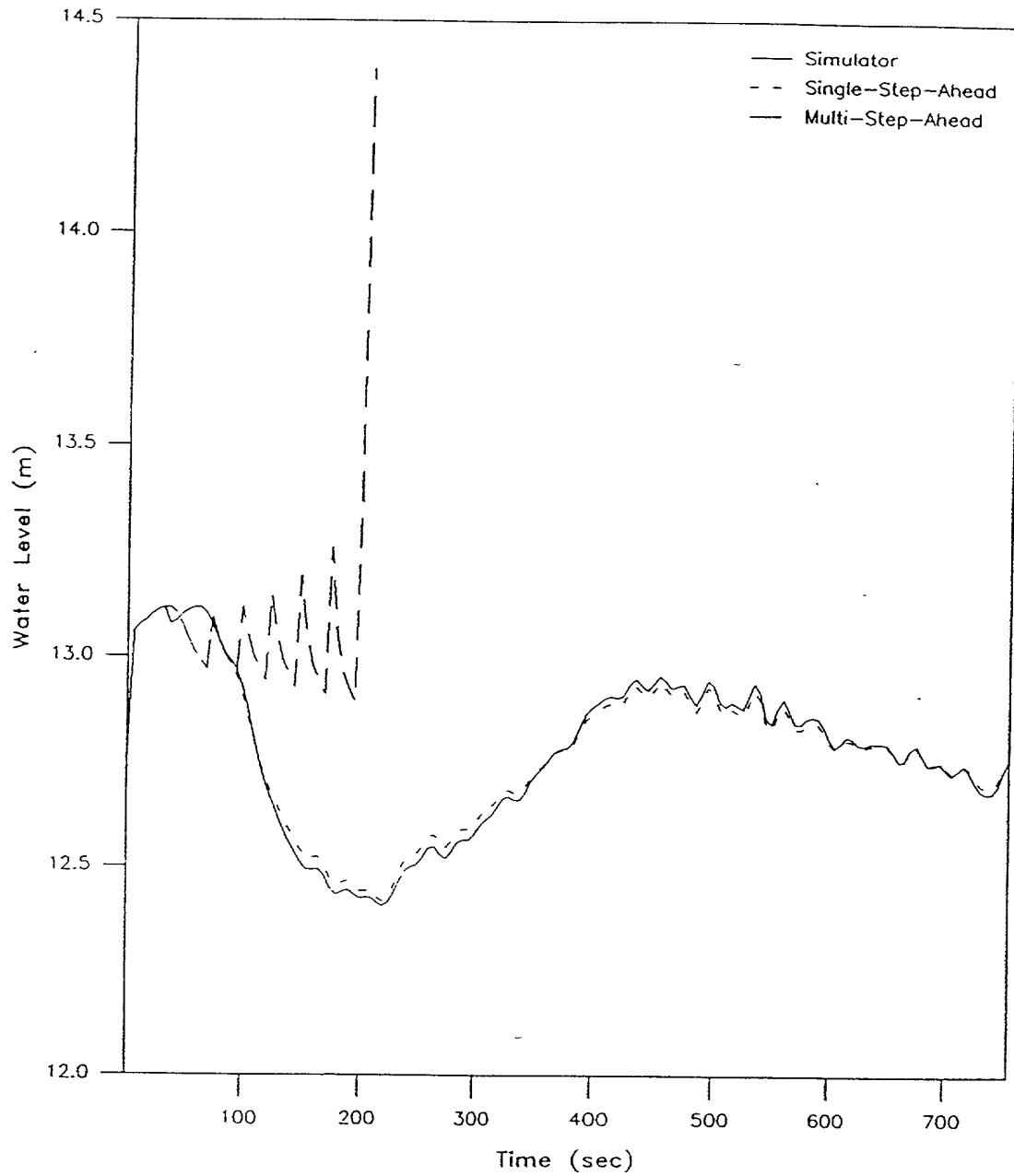


Figure 41. Polynomial NARX Water Level Response to a Step Power Level Increase from 20% to 25% of Full Power.

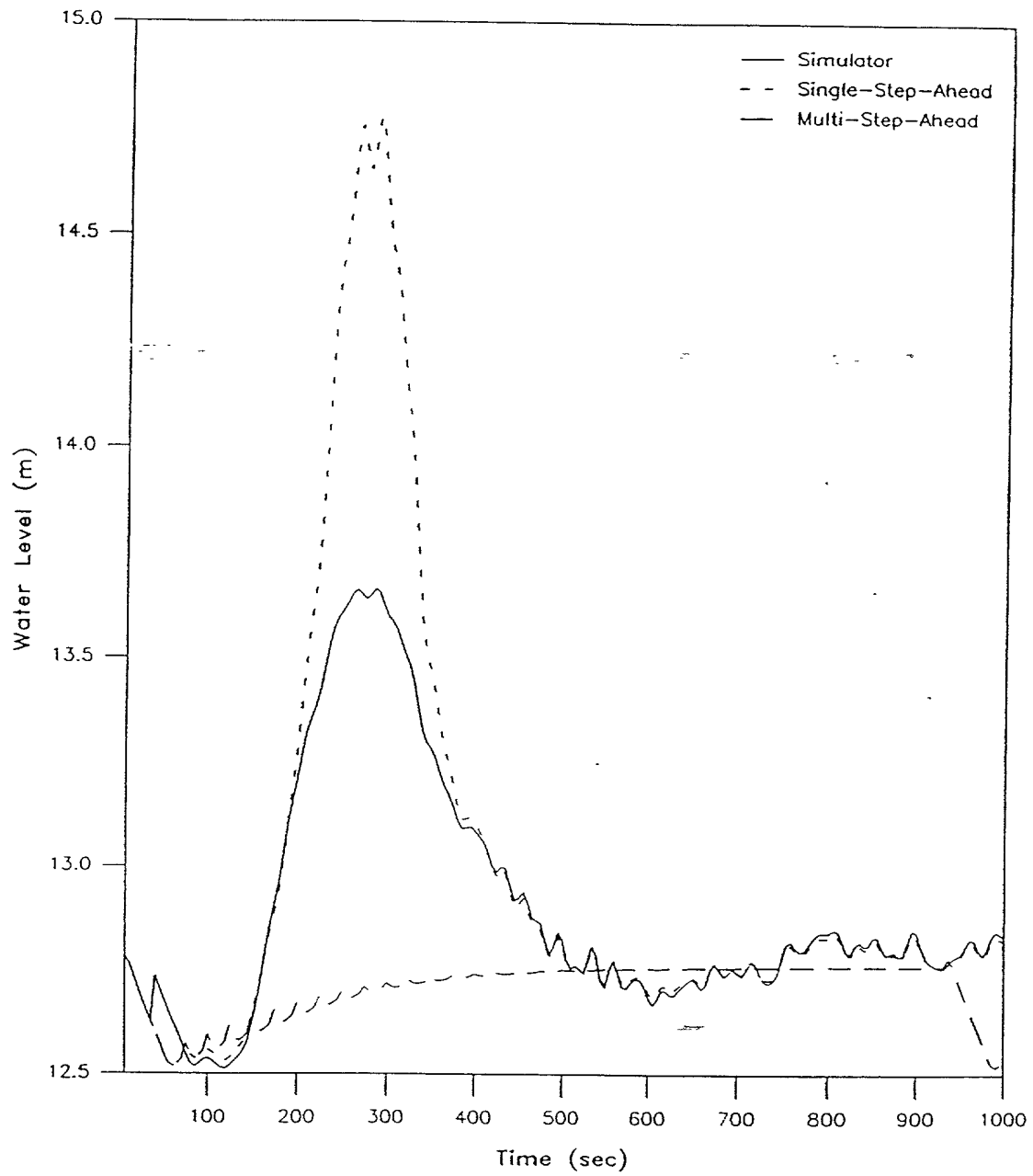


Figure 42. Polynomial NARX Water Level Response to a Ramp Power Level Decrease from 35% to 20% of Full Power.

V.5.2 Computational Neural Networks Model Using Teacher Forcing

In this method the individual weight update mode is used because it gave better results than the batch weight update mode in the case study. Again, similarly to the case study two data sets were used in the predictor design: the training data set and the testing data set. The CNN was trained with the training data set using TF, but the weights are chosen based on the network performance on the testing data set using GF. Training is stopped when the Mean Square Error (MSE) on the testing data set starts to increase, thus preventing overfitting.

The medium power level range for the water level was modeled. The objective is to design a network capable of accurate MSP. The learning rate chosen is $\eta = 0.001$ with one delayed output utilized. A node search was performed using the guidelines given in chapter III. The neural network which performed the best is a 4-3-2-1 network. Its performance was much better than the performance of the polynomial NARX method in both SSP and MSP. However, it failed to accurately predict the water level response for many step and ramp power level increases within the training set and the validation set when considering MSP. Figures 43 and 44 give the water level response to a step power level increase from 20% to 25% of full power, and ramp decrease from 35% of full power level to 20% of full power level for SSP and MSP, respectively. The UTSG simulator water level response and the CNN predictor water level response are both shown in these figures. As it can be seen from these figures, the SSP water level response of the CNN predictor is perfect for both transients. However, the MSP water level of the CNN predictor response is not accurate enough. An attempt has been

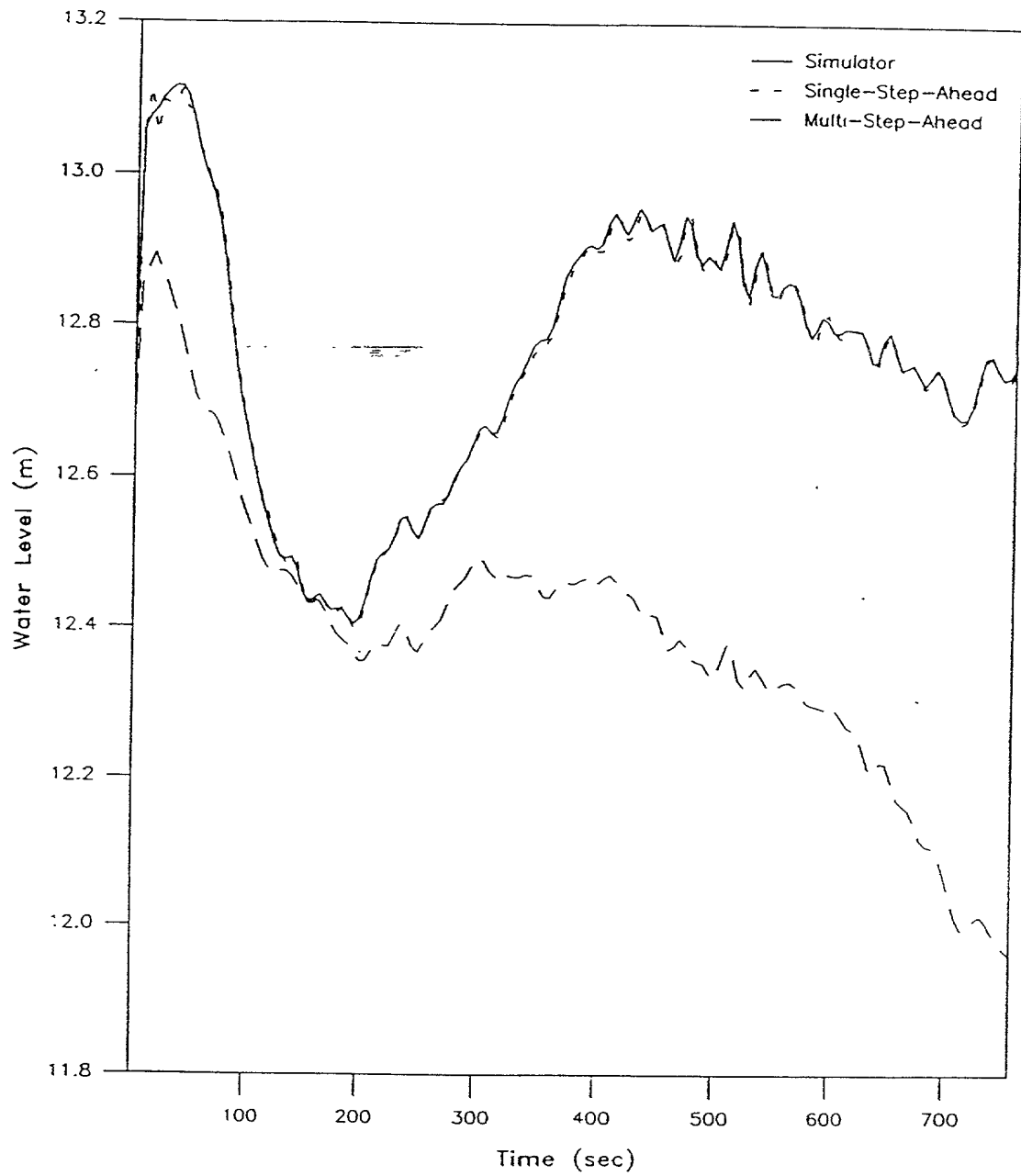


Figure 43. Teacher Forcing CNN Water Level Response to a Step Power Level Increase from 20% to 25% of Full Power.

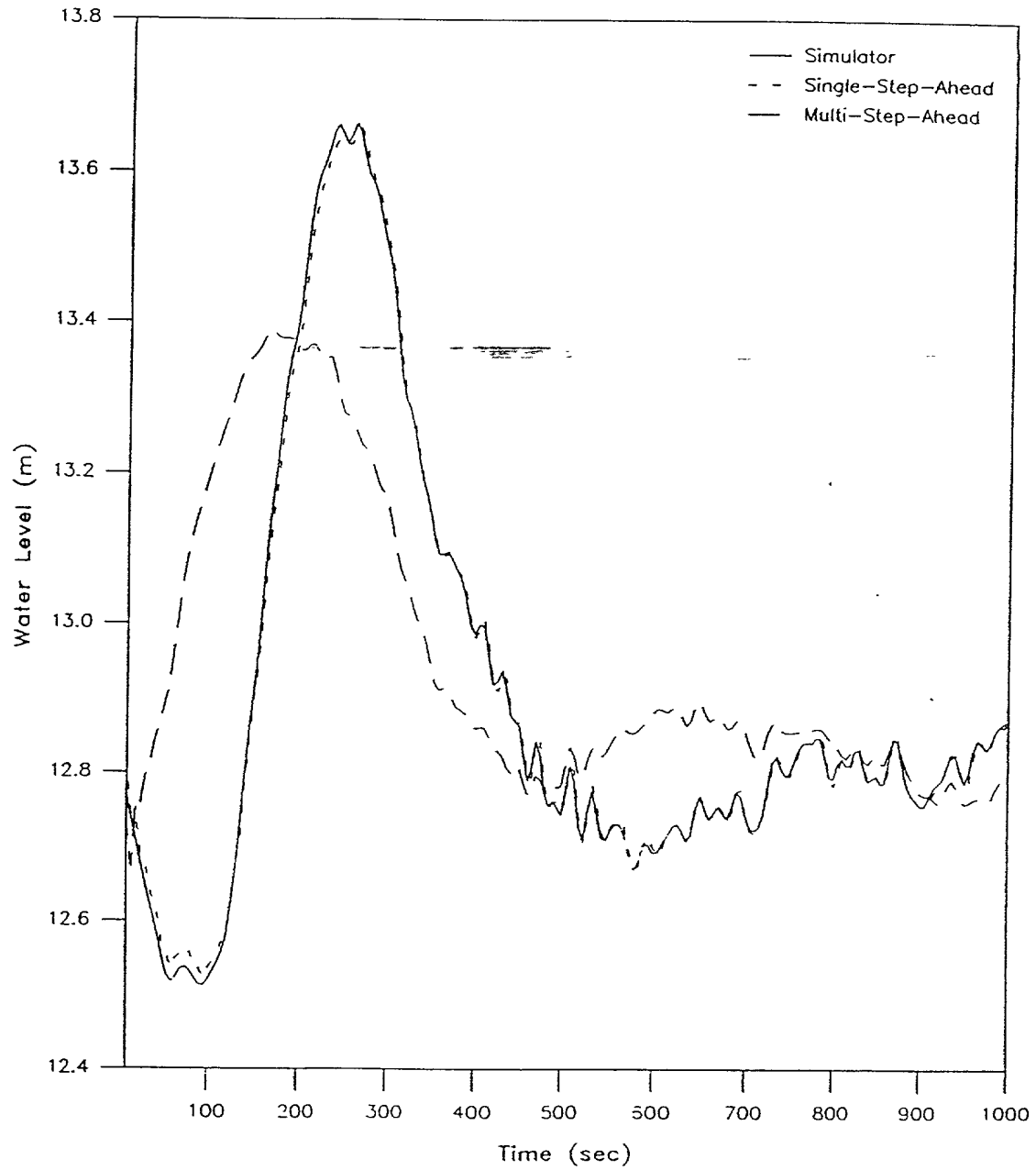


Figure 44. Teacher Forcing CNN Water Level Response to a Ramp Power Level Decrease from 35% to 20% of Full Power.

made to utilize more past delayed outputs, as well as more delayed inputs, namely the feedwater flow rate and the steam flow rate, but no significant MSP accuracy improvement was observed. The performance of the TF method is still much better than the performance of the polynomial NARX conventional method for both in SSP and MSP. Note that it takes at least one hour of computer time to generate each transient using the simulator, as compared to few seconds using the TF CNN predictor. This is of great importance in the area of control engineering and fault-diagnosis, because it allows the user to know ahead of time the future behavior of the system so that appropriate corrective action can be taken.

The reason for the partial failure of the CNN in MSP using the TF method was further investigated by generating the same data set generated previously, but this time without process or sensors noise. Again, a node search was performed for the best network architecture. Following removal of the noise in the training, the CNN was capable of performing MSP. This was done for analysis purposes only, since all data associated with the operation of the UTSG are corrupted with significant colored noise. To prove this point further, an attempt was made to model the high power level range of the UTSG which has a relatively higher signal to noise ratio (SNR) than the medium power level range. The obtained CNN predictor using TF gave satisfactory results, however, as it will be shown in the next subsection, the GF method gave much better results. These two separate investigations clearly demonstrate that noise is a major factor into the design of CNN predictor capable of accurate MSP

V.5.3 Computational Neural Networks Model Using Global Feedback

In using the GF method the individual weight update mode is used because it gave better results than the batch weight update mode in the case study. Similarly to the TF method two data sets were used: the training data set and the testing data set. The CNN is trained on the training data set using GF, and the weights are chosen based on the network performance in the testing data set using GF. Training is stopped when the MSE on the testing data set starts to increase. Before performing a node search, a seed search was performed with a learning rate of $\rho = 0.0005$. The seed search was a critical step since the network was not able to learn any of the training data set without the appropriate weight and bias initialization.

Using the procedures outlined in chapter III, it was determined that one delayed output and two delayed inputs were to be utilized. Note that only two of the four inputs were delayed for the use in the network, namely, the feedwater flow rate and the steam flow rate. Figure 45 gives a graphical presentation of the CNN predictor adopted for use in modeling the UTSG. In this figure k is the time step, $u_1(k-1)$ is the feedwater flow rate, $u_2(k-1)$ is the steam flow rate, $u_3(k-1)$ is the feedwater flow rate temperature, and $u_4(k-1)$ is the hot-leg temperature. The predicted output $\hat{y}(k/k-1)$ is either the water level or the steam pressure of the secondary side of the UTSG. The search for the number of hidden nodes converged to four nodes in a

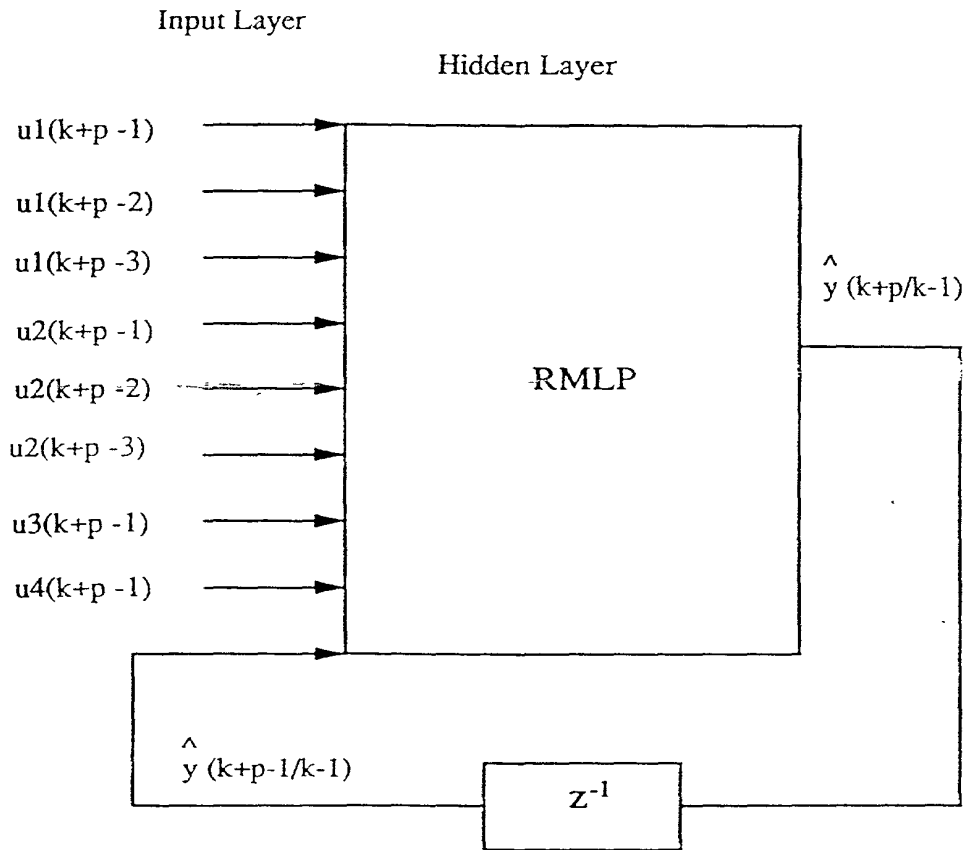


Figure 45. Computational Neural Network Predictor Used for Multi-Step-Ahead Prediction.

single hidden layer for all the six CNN predictors designed for modeling the UTSG.

The obtained CNN predictors using the GF method give good performance for both MSP and SSP at all power level ranges. To confirm this finding extensive new

simulation tests have been performed for validating the UTSG empirical model at all power levels. Figures 46 through 76 give the water level response and the secondary pressure response to different transients using the obtained CNN predictors and the UTSG simulator. Similarly to the TF method, it takes at least one hour of computer time to generate each transient using the simulator. whereas it only takes few seconds for the GF CNN predictor to complete the transient.

Modeling the low power range was a more difficult task than modeling the medium and high power ranges. This is because at the low power range the noise to signal ratio is very high compared to the medium and high power level ranges. As it was shown earlier, noise is a major issue in designing CNN predictors. The stopping criterion was reached after 11000 iterations for the water level predictor, and 14000 iterations for the steam pressure predictor. The number of iterations needed to reach the stopping criterion depends on the weight and bias initialization, which are currently initialized randomly. If the initialization is good, then the number of iterations required to reach the stopping criterion will be reduced. On the other hand a bad initialization set may require more iterations in order to reach the stopping criterion. Figure 46 through 55 give the simulations at the low power range, while Table 4 gives the Relative Mean Square Error (RMSE) and the maximum relative error encountered during the transients for SSP and MSP. As it can be seen from these figures, the performance of the obtained CNN predictor at low power levels is not as good for the water level response as is for the saturation pressure. This is because of the high level process and sensor noise, and also because of the nonminimum phase effects which are more

Table 4. UTSG Empirical Model Validation at Low Power Levels.

Test Case	SSP	MSP
Water Level	RMSE [Max R.Error]%	RMSE [Max R.Error]%
5% to 7.5% Step	1.81 [1.89]	1.90 [1.96]
7.5% to 10% Step	1.38 [1.80]	1.51 [1.80]
10% to 15% Step	1.52 [1.17]	1.63 [1.34]
15% to 20% Ramp	1.40 [2.02]	1.41 [1.60]
5% to 20% Ramp	1.70 [1.87]	1.83 [2.58]
20% to 5% Ramp	1.86 [2.59]	1.99 [2.96]
5% to 15% Ramp	1.87 [1.86]	2.06 [2.57]
15% to 5% Ramp	1.68 [2.06]	1.80 [2.00]
10% to 20% Ramp	1.74 [1.71]	2.04 [2.32]
20% to 10% Ramp	1.61 [2.60]	1.83 [1.98]
Saturation Pressure	RMSE [Max R.Error]%	RMSE [Max R.Error]%
5% to 7.5% Step	0.065 [0.37]	0.18 [0.61]
7.5% to 10% Step	0.076 [0.37]	0.20 [0.60]
10% to 15% Step	0.079 [0.49]	0.30 [0.55]
15% to 20% Ramp	0.096 [0.53]	0.48 [0.77]
5% to 20% Ramp	0.085 [0.70]	0.43 [0.83]
20% to 5% Ramp	0.064 [0.60]	0.21 [0.81]
5% to 15% Ramp	0.073 [0.71]	0.36 [0.83]
15% to 5% Ramp	0.063 [0.21]	0.21 [0.67]
10% to 20% Ramp	0.075 [0.35]	0.46 [0.63]
20% to 10% Ramp	0.062 [0.57]	0.28 [0.81]

pronounced at low power levels. Utilizing additional delayed outputs and delayed inputs did not improve these results.

The models obtained for the medium power range gave good performance throughout all the simulated transients. The stopping criterion was reached after 11000 iterations for the water level response predictor, and 2500 iterations for the steam pressure response predictor. Figure 56 through 66 give the simulations of the medium power levels for SSP and MSP. Table 5 gives the Relative Mean Square Error (RMSE) and the maximum relative error during the transient for SSP and MSP. The prediction

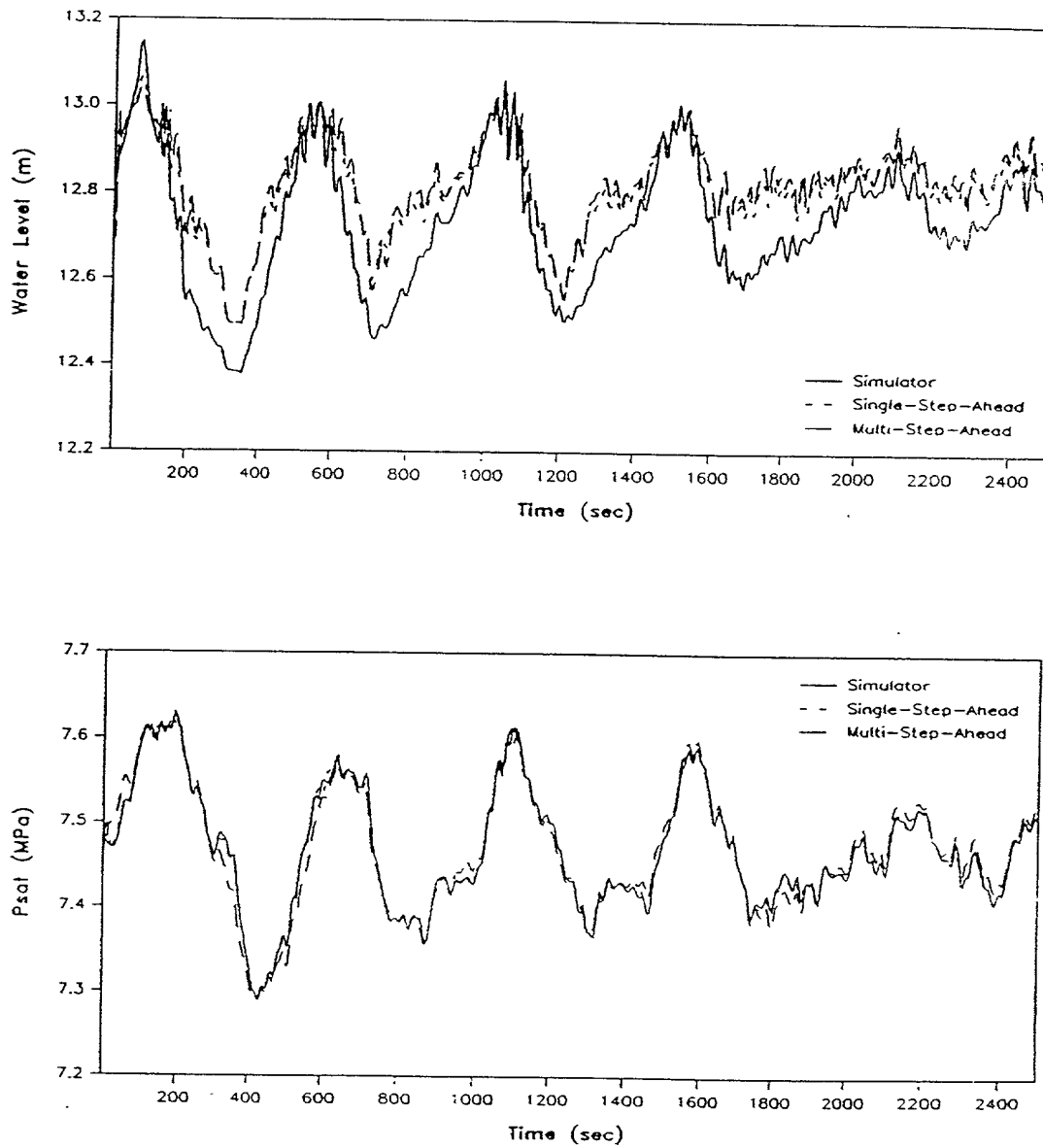


Figure 46. UTSG Transient Response Prediction for a 5% to 7.5% of Full Power Step.

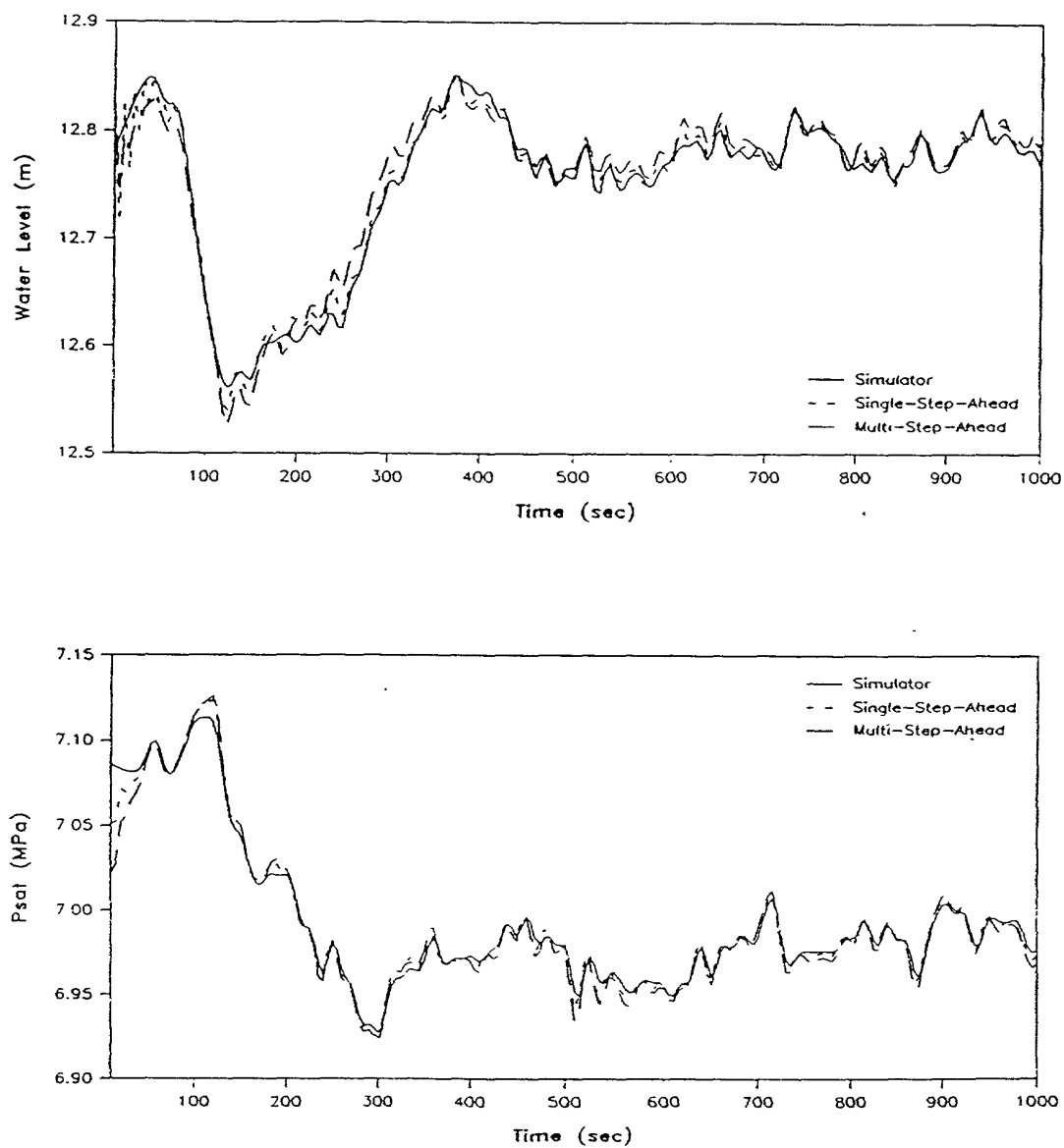


Figure 47. UTSG Transient Response Prediction for a 7.5% to 10% of Full Power Step.

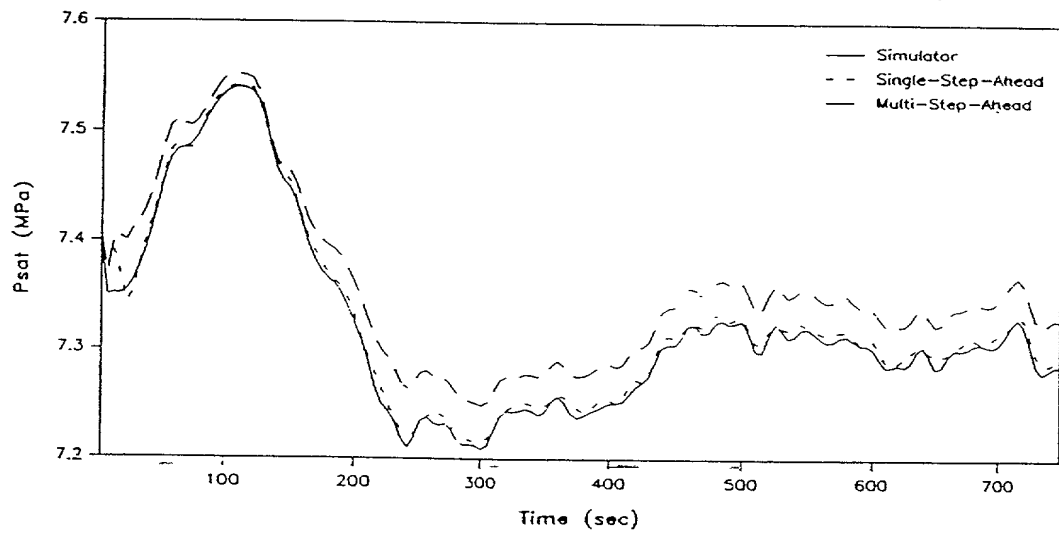
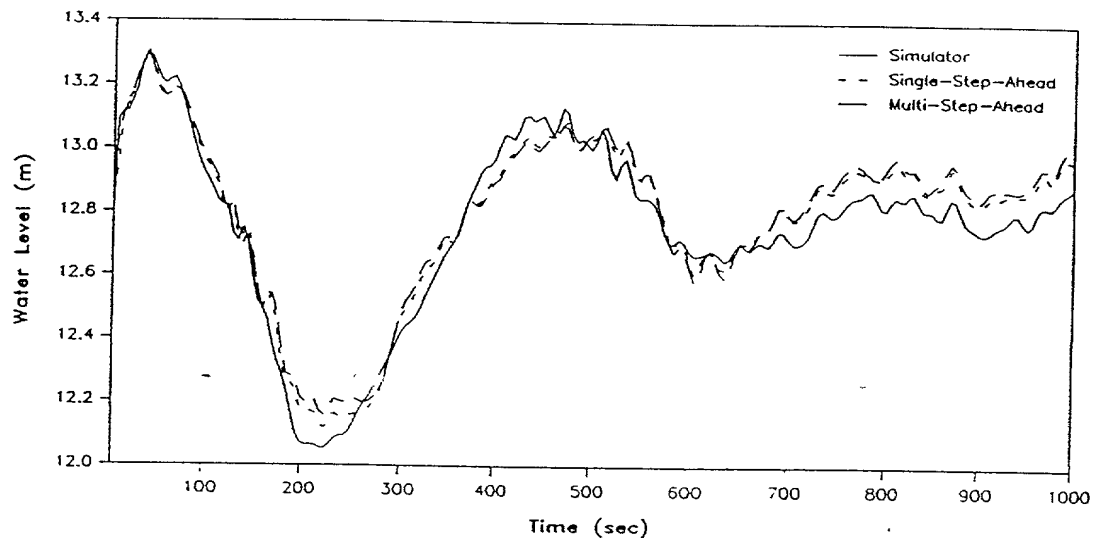


Figure 48. UTSG Transient Response Prediction for a 10% to 15% of Full Power Step.

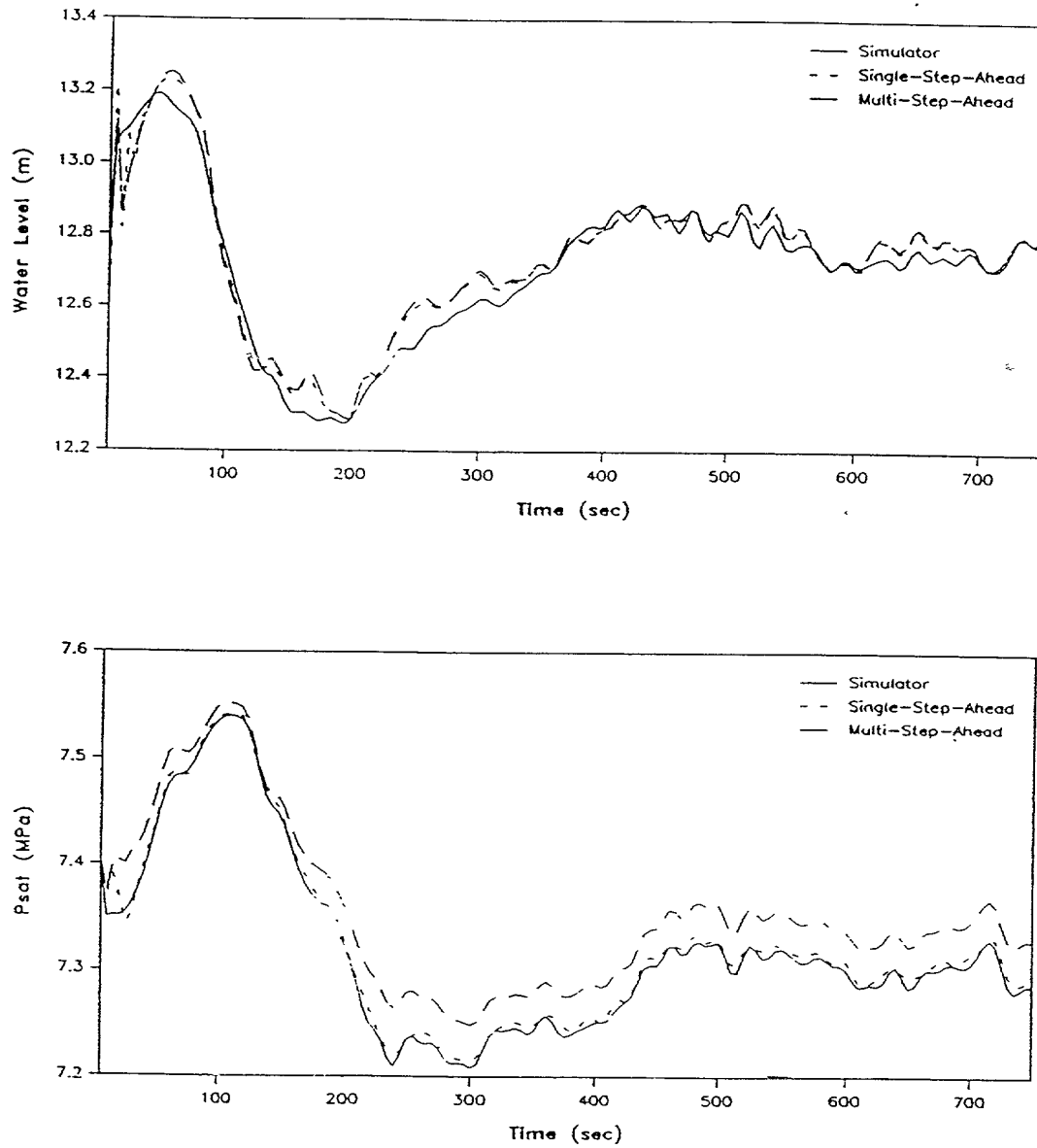


Figure 49. UTSG Transient Response Prediction for a 15% to 20% of Full Power Step.

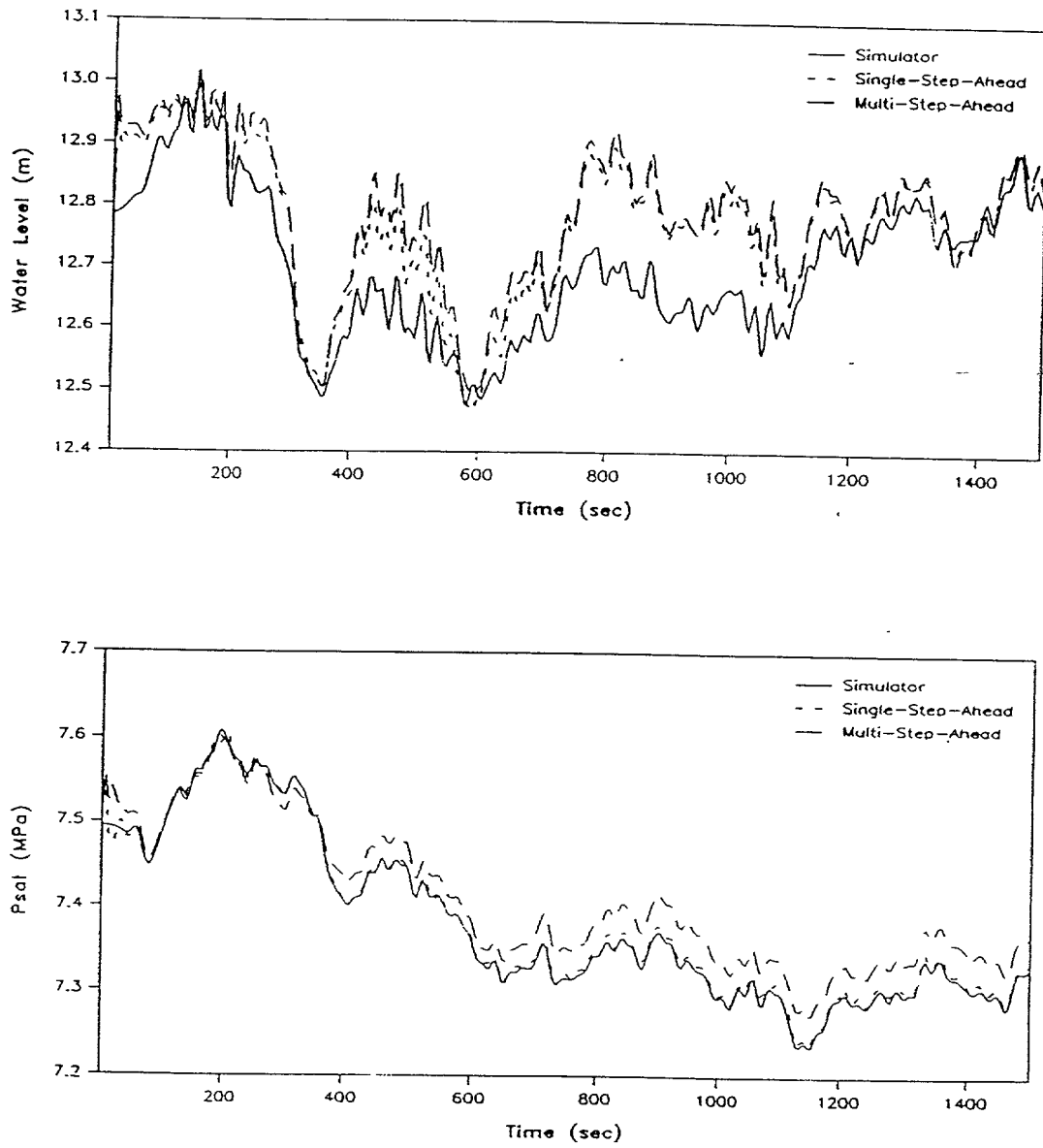


Figure 50. UTSG Transient Response Prediction for a 5% to 20% of Full Power Ramp.

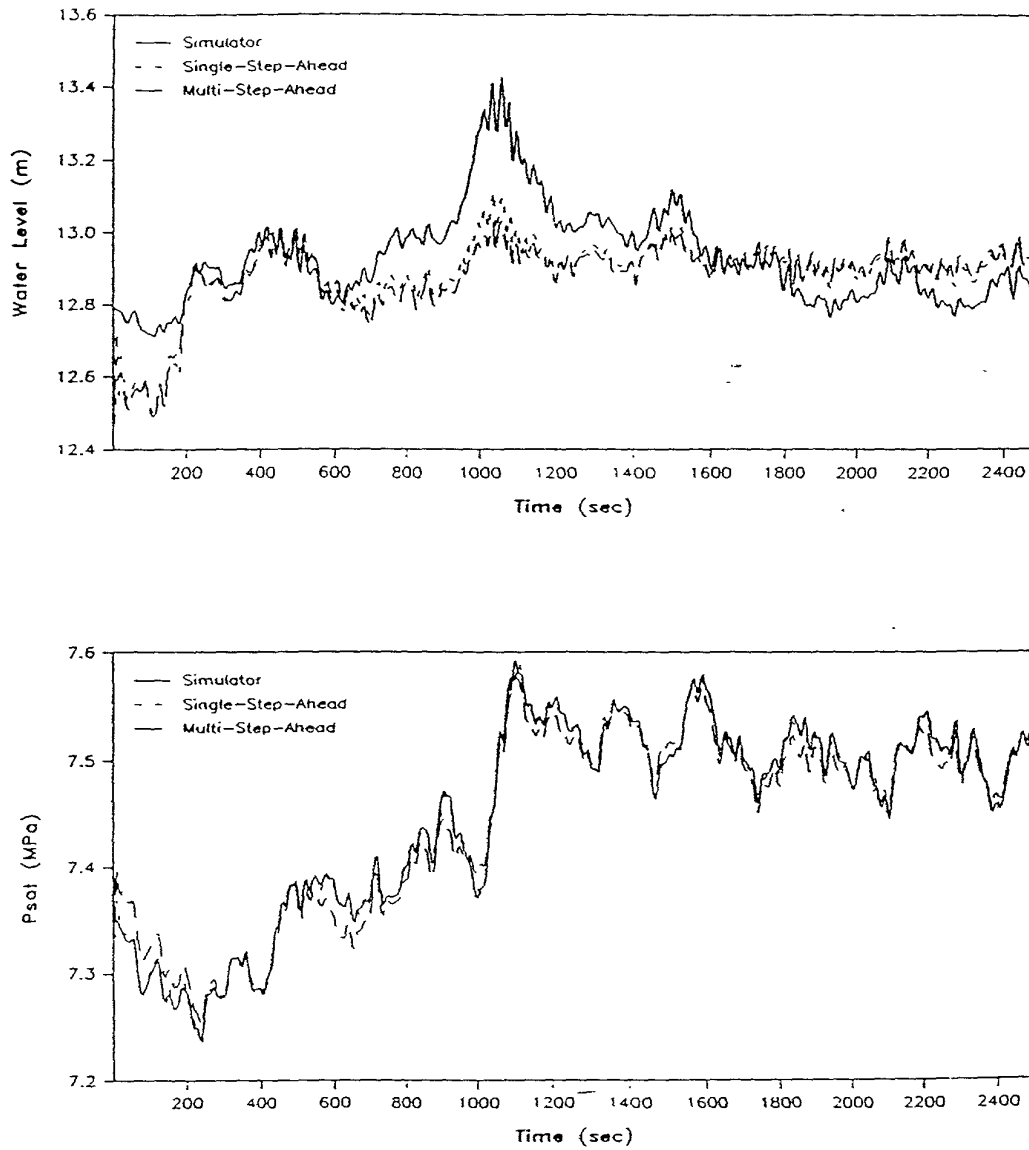


Figure 51. UTSG Transient Response Prediction for a 20% to 5% of Full Power Ramp.

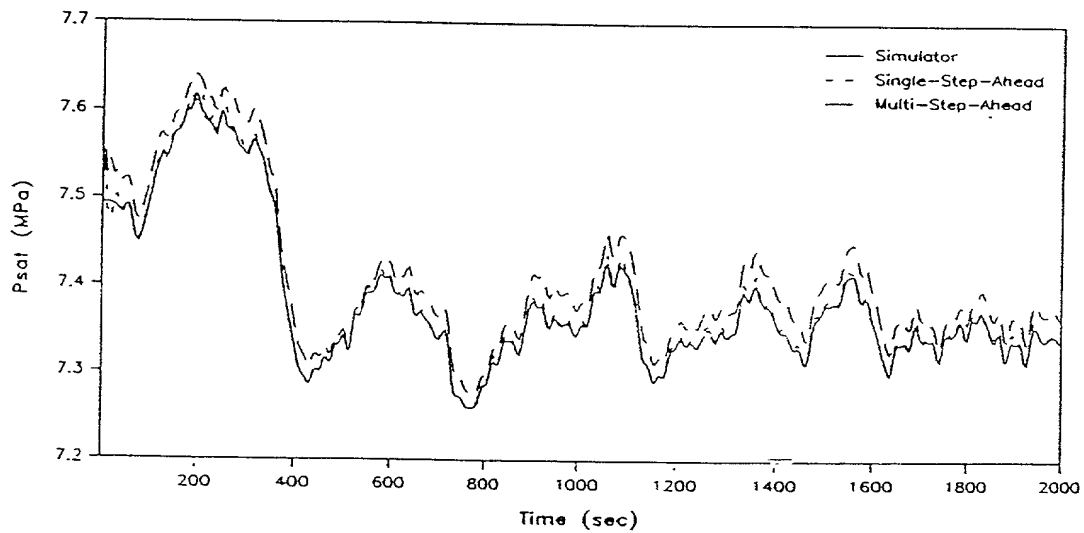
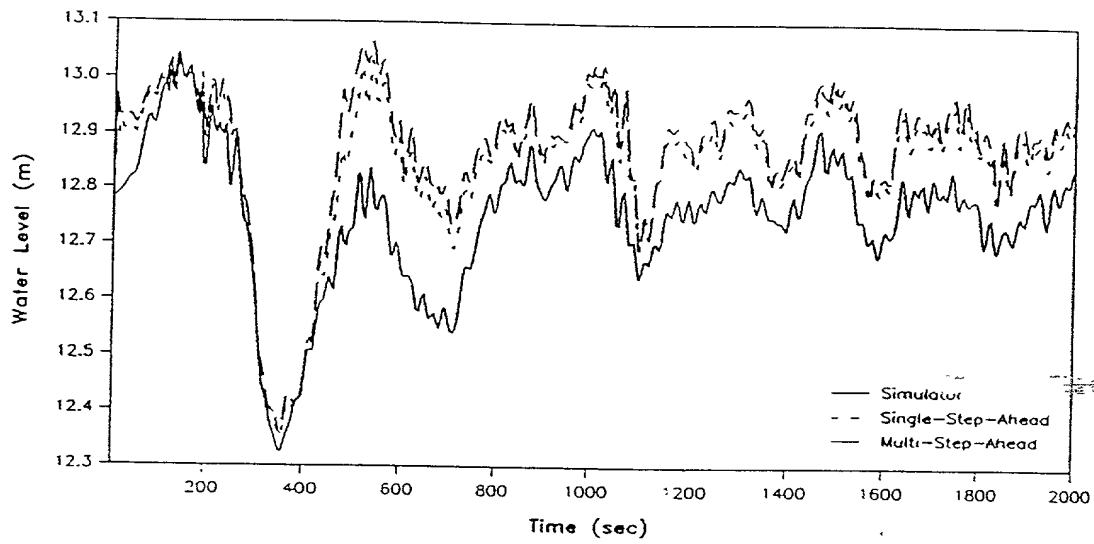


Figure 52. UTSG Transient Response Prediction for a 5% to 15% of Full Power Ramp.

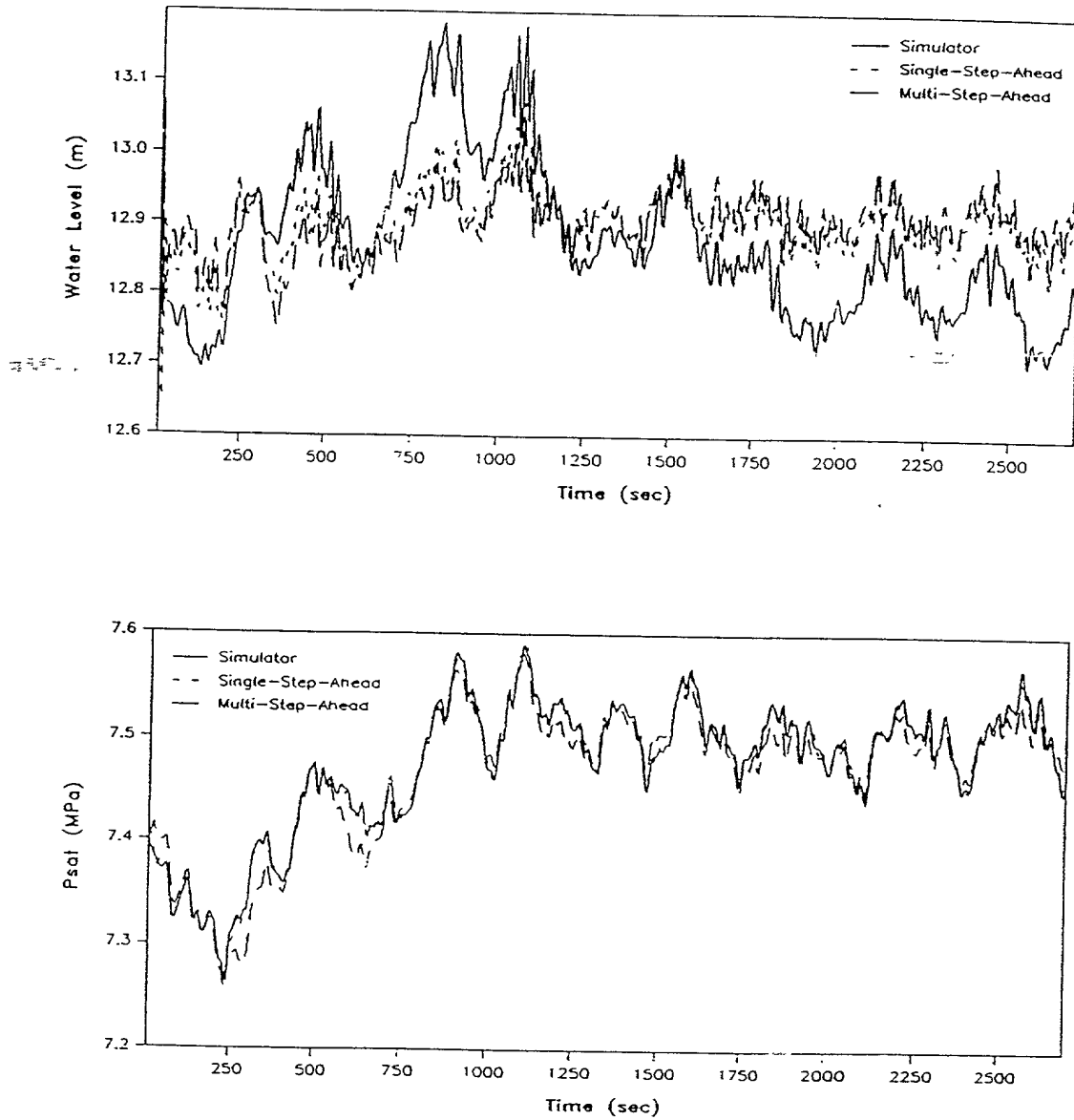


Figure 53. UTSG Transient Response Prediction for a 15% to 5% of Full Power Ramp.

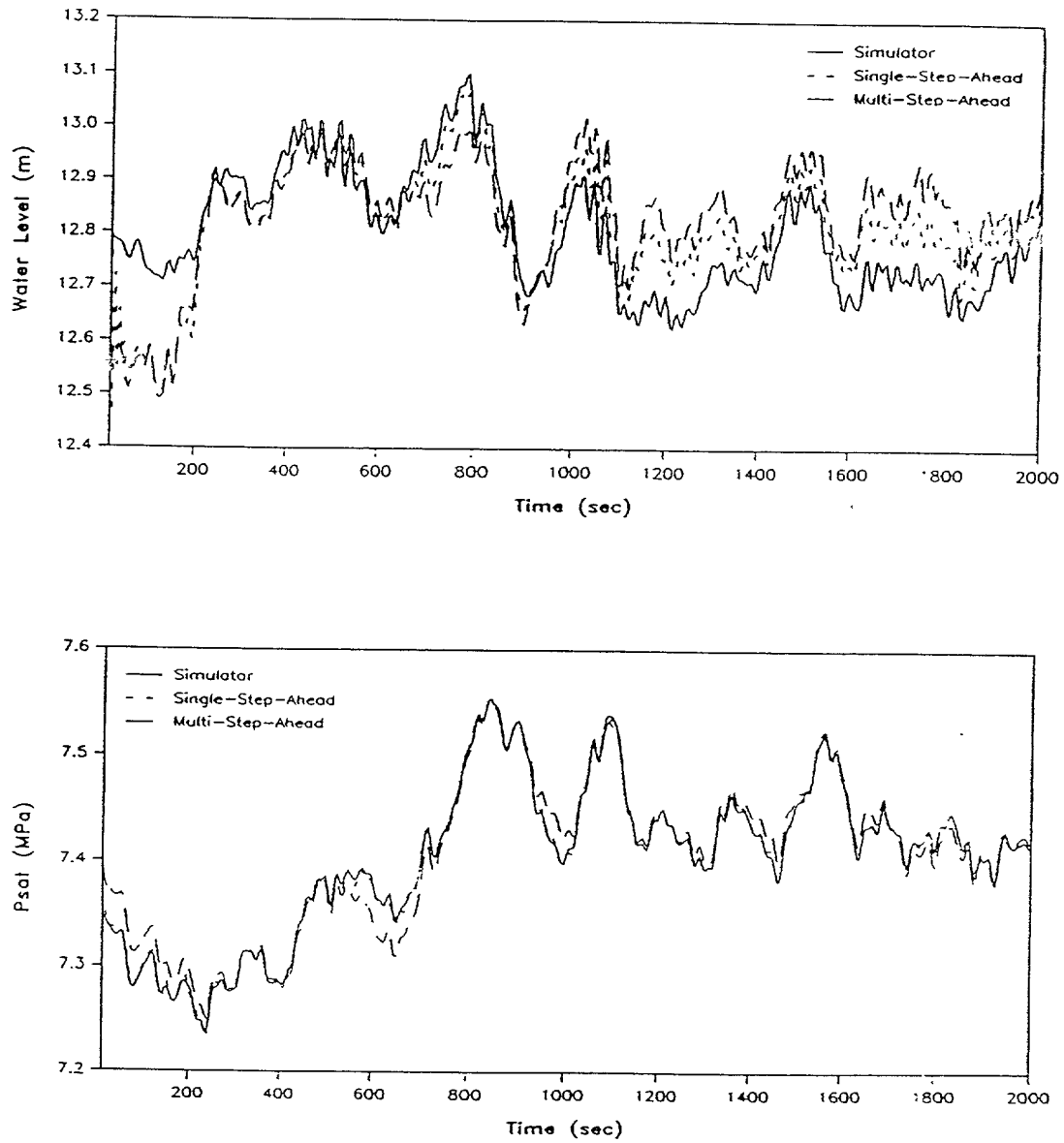


Figure 54. UTSG Transient Response Prediction for a 20% to 10% of Full Power Ramp.

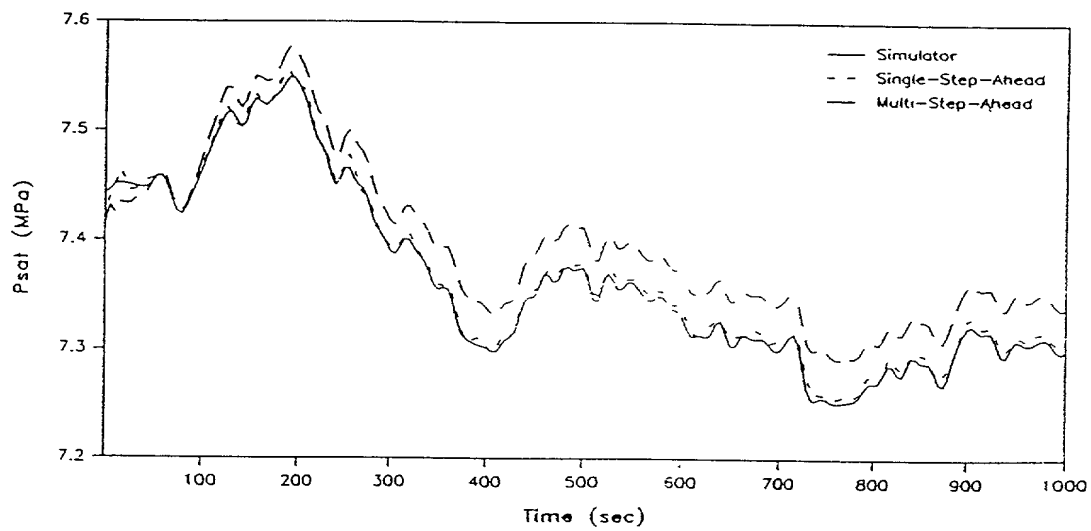
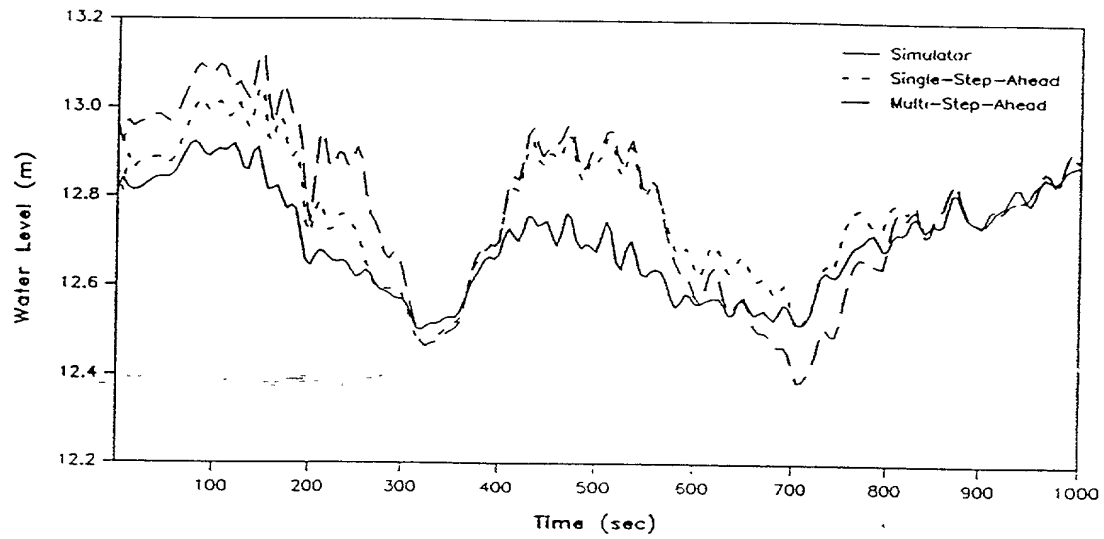


Figure 55. UTSG Transient Response Prediction for a 10% to 20% of Full Power Ramp.

Table 5. UTSG Empirical Model Validation at Medium Power Levels.

Test Case	SSP	MSP
Water Level	RMSE [Max R.Error]%	RMSE [Max R.Error]%
20% to 25% Step	0.21 [0.67]	0.59 [1.16]
30% to 35% Step	0.26 [1.36]	0.31 [1.26]
40% to 45% Step	0.14 [1.40]	0.28 [1.26]
20% to 50% Ramp	0.27 [3.20]	0.59 [3.63]
50% to 20% Ramp	0.18 [0.92]	0.30 [0.88]
35% to 45% Ramp	0.37 [3.60]	0.70 [3.44]
45% to 35% Ramp	0.23 [0.82]	0.72 [2.29]
20% to 35% Ramp	0.33 [3.12]	0.57 [3.54]
35% to 20% Ramp	0.24 [1.02]	0.96 [3.27]
20% to 30% Ramp	0.38 [3.13]	0.63 [3.55]
30% to 20% Ramp	0.26 [1.14]	1.21 [3.73]
Saturation Pressure	RMSE [Max R.Error]%	RMSE [Max R.Error]%
20% to 25% Step	0.10 [0.53]	0.15 [0.94]
30% to 35% Step	0.09 [0.30]	0.10 [0.31]
40% to 45% Step	0.12 [1.10]	0.17 [1.10]
20% to 50% Ramp	0.13 [0.83]	0.27 [0.96]
50% to 20% Ramp	0.14 [1.58]	0.26 [1.59]
35% to 45% Ramp	0.11 [1.04]	0.12 [1.03]
45% to 35% Ramp	0.11 [0.35]	0.30 [0.58]
20% to 35% Ramp	0.15 [0.95]	0.21 [1.04]
35% to 20% Ramp	0.16 [0.78]	0.47 [1.12]
20% to 30% Ramp	0.17 [0.92]	0.32 [0.99]
30% to 20% Ramp	0.16 [0.52]	0.49 [0.76]

for the steam pressure are better than the prediction for the water level. This is because the secondary steam pressure is not subject to the nonminimum phase effects. Note that the RMSE of the simulated transients at medium power levels are lower than the ones for the low power levels. This was expected because the SNR at medium power levels is four times higher than the SNR at low power levels, and the SNR at high power levels is about 13 times higher than the SNR at low power levels. Another point to notice is that the SSP using the GF is not as good as the SSP using the

TF. This can be seen by comparing Figures 43 and 56. The same results can also be seen by comparing Figures 44 and 64. These results were expected as the CNN predictors were solely designed for the purpose of MSP.

The predictors obtained for the high power range also gave good performance through-out the simulated transients. The stopping criterion was reached after 17500 iterations for the water level response predictor, and 3500 iterations for the steam pressure response predictor. Figures 67 through 76 give the simulations of the high power levels for SSP and MSP. Table 6 gives the Relative Mean Square Error (RMSE) and the maximum relative error during the transients for both the SSP and MSP. The responses for the steam pressure are as good as the responses for the water level. This is because the nonminimum phase effects are not significant at higher power levels. Note that RMSE of the simulated transients at high power levels are lower than the ones of the medium and low power levels. This was expected as the SNR at high power levels is much higher than the SNR at medium and low power levels.

Finally, in order to test the performance limits of the UTSG CNN predictors, a fast controlled ramp load drop from 100% to 5% of full power was simulated. The 5% of full power level was to be reached after 700 sec. The results of this test are shown in Figure 77. As it can be seen from this figure, the performance of the CNN predictor started deteriorating at approximately 500 sec for the MSP, long before the controller started failing to effectively control the system, which occurred at approximately 620 sec, while the SSP water level response is satisfactory. To further test the performance limits of the CNN predictors, two open-loop fast ramp drops were also simulated.

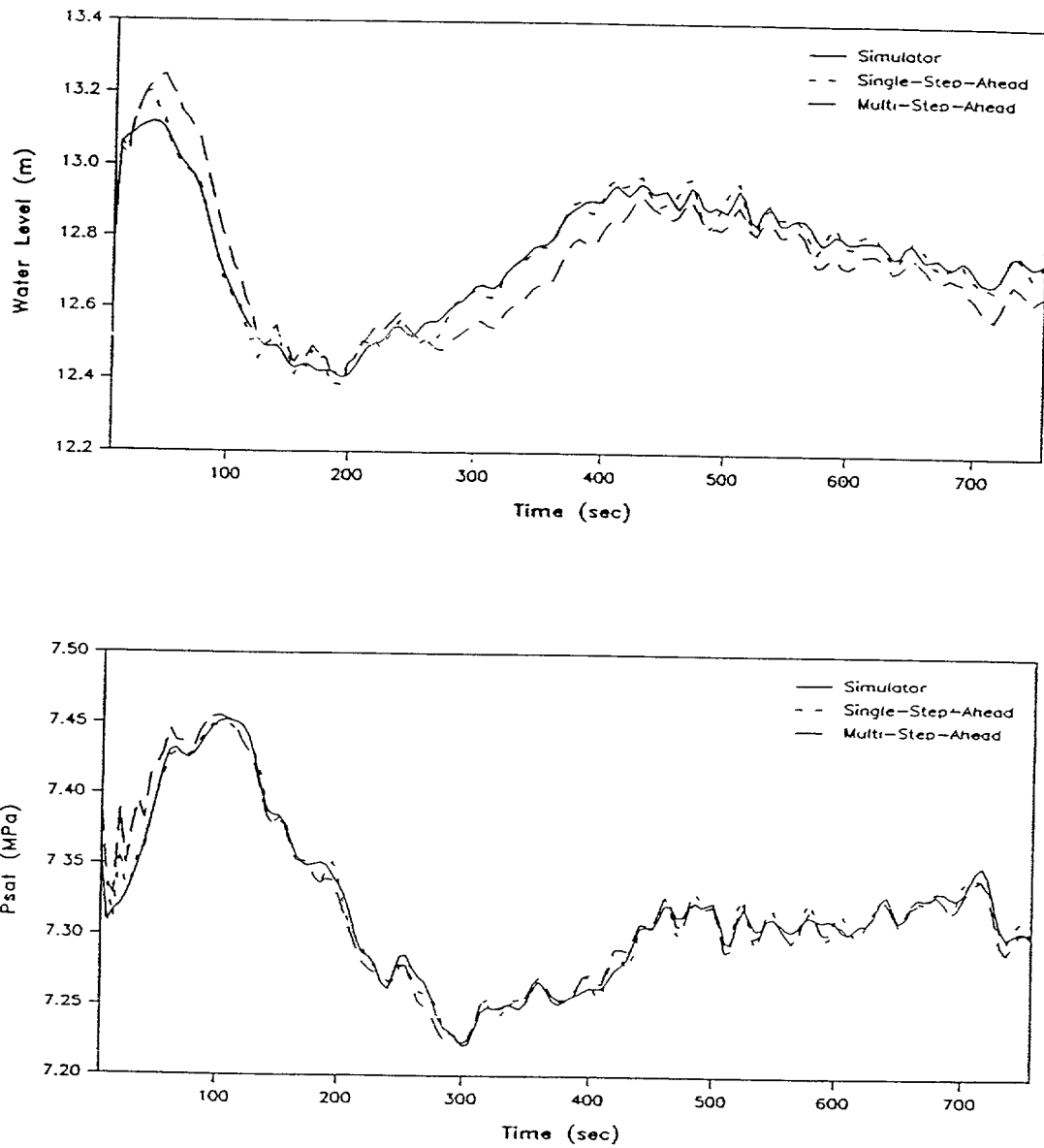


Figure 56. UTSG Transient Response Prediction for a 20% to 25% of Full Power Step.

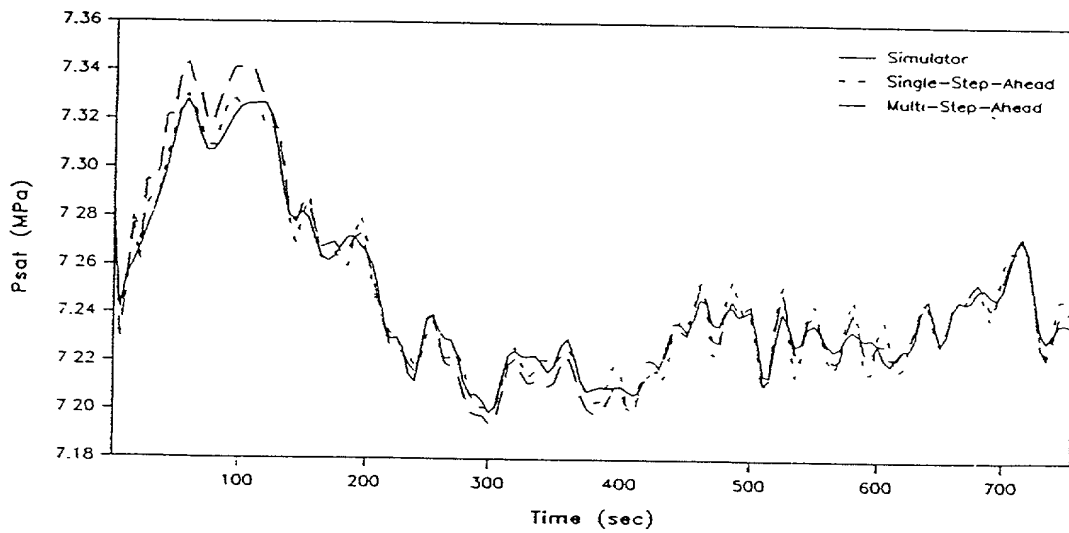
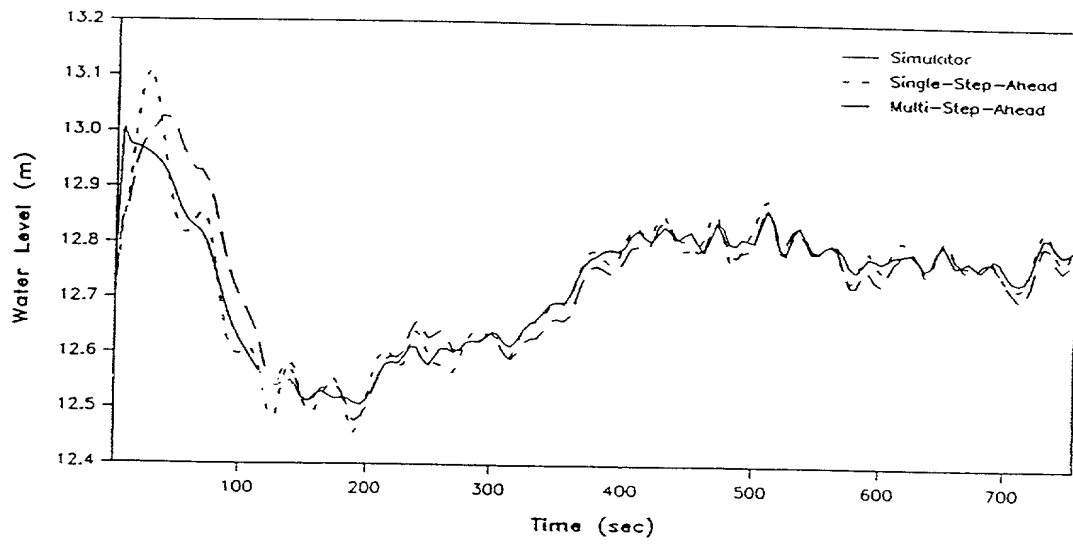


Figure 57. UTSG Transient Response Prediction for a 30% to 35% of Full Power Step.

226

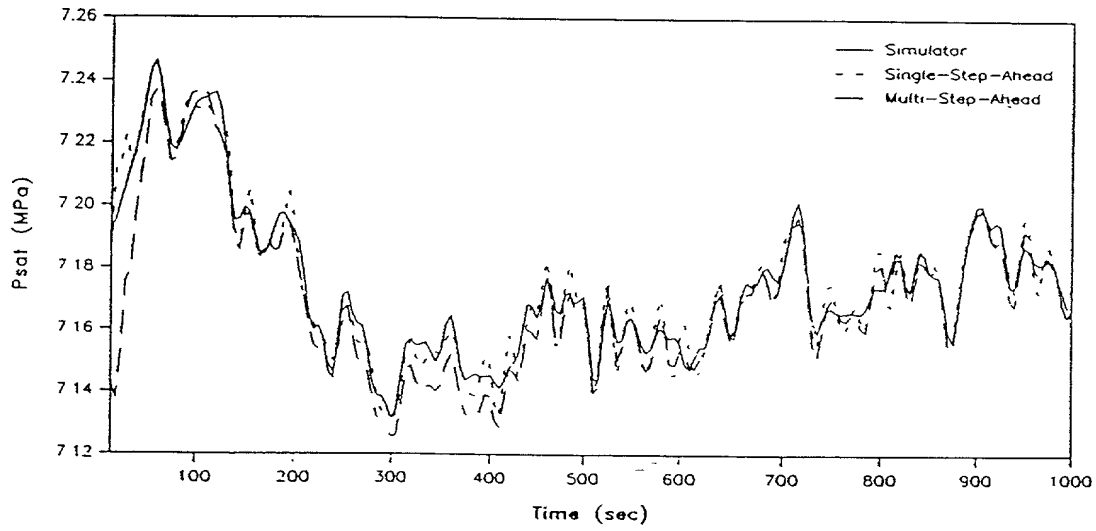
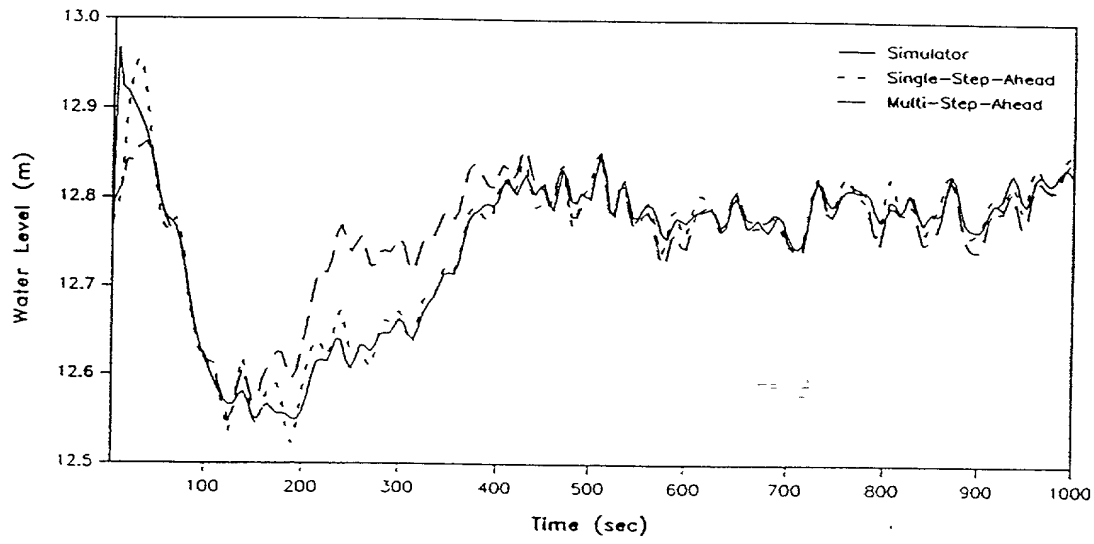


Figure 58. UTSG Transient Response Prediction for a 40% to 45% of Full Power Step.

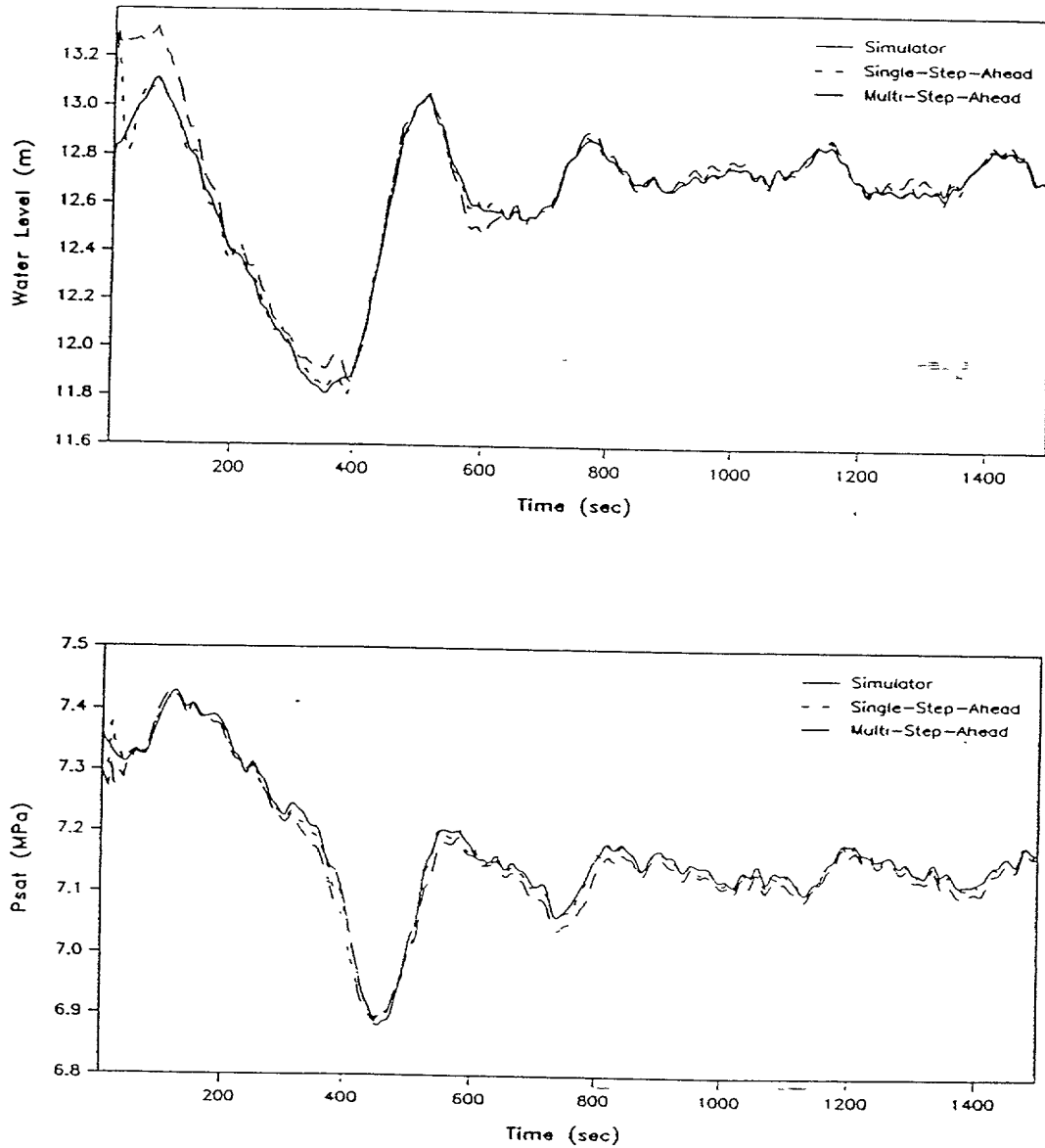


Figure 59. UTSG Transient Response Prediction for a 20% to 50% of Full Power Ramp.

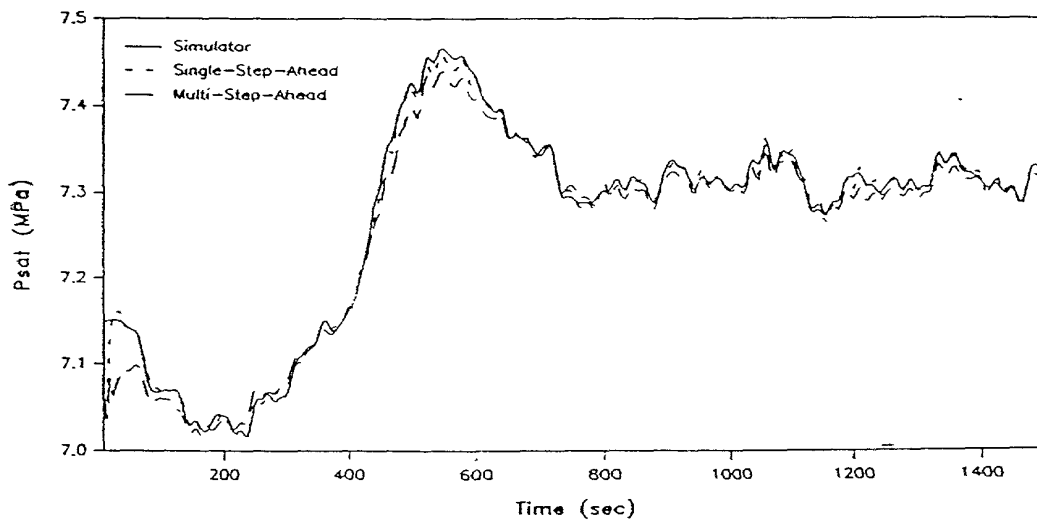
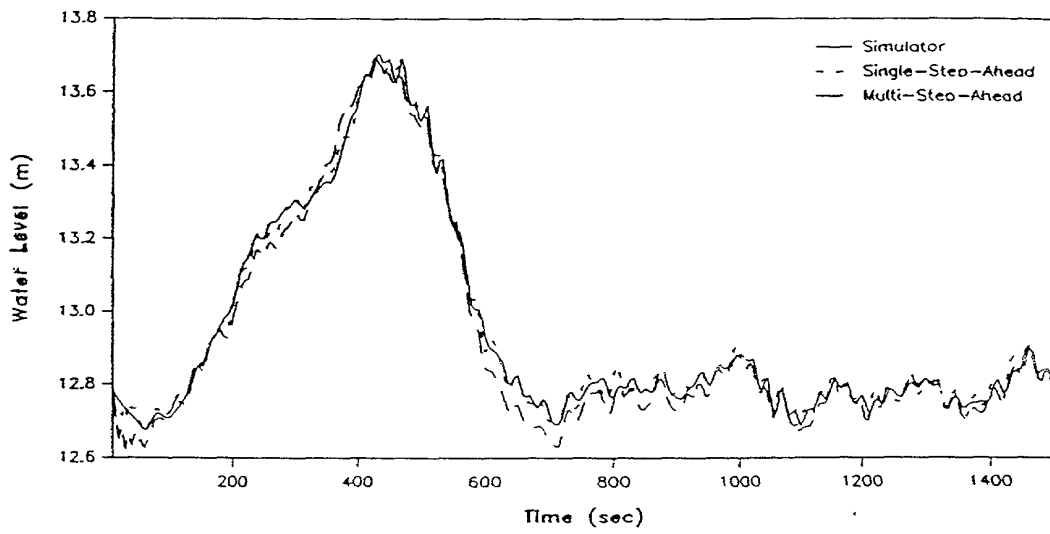


Figure 60. UTSG Transient Response Prediction for a 50% to 20% of Full Power Ramp.

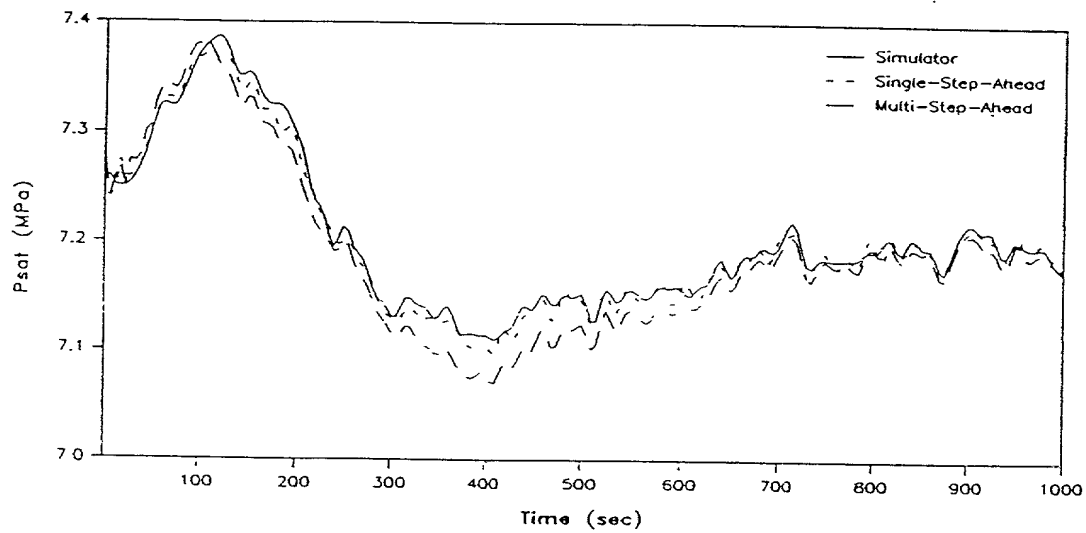
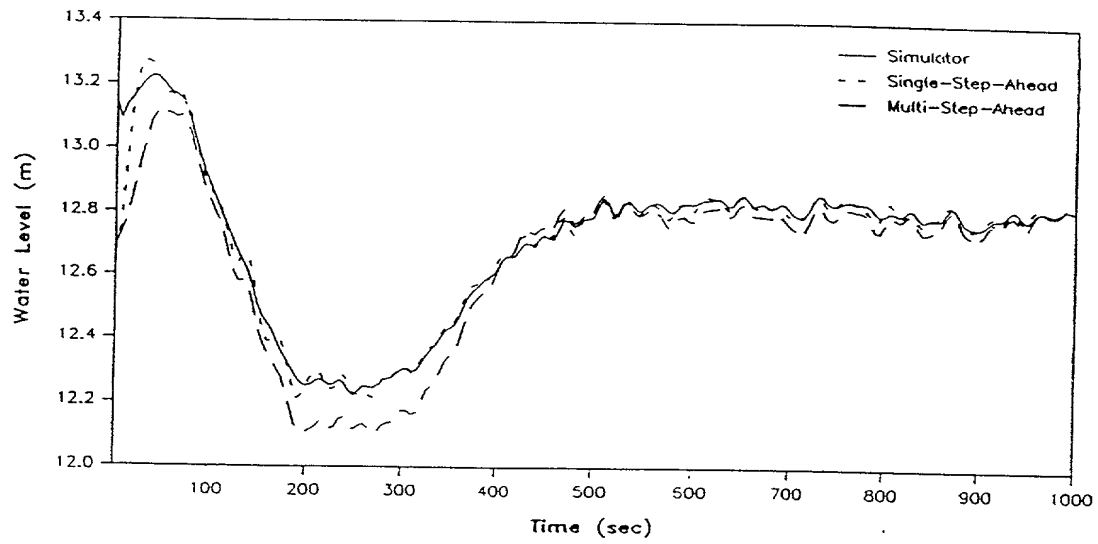


Figure 61. UTSG Transient Response Prediction for a 35% to 45% of Full Power Ramp.

230

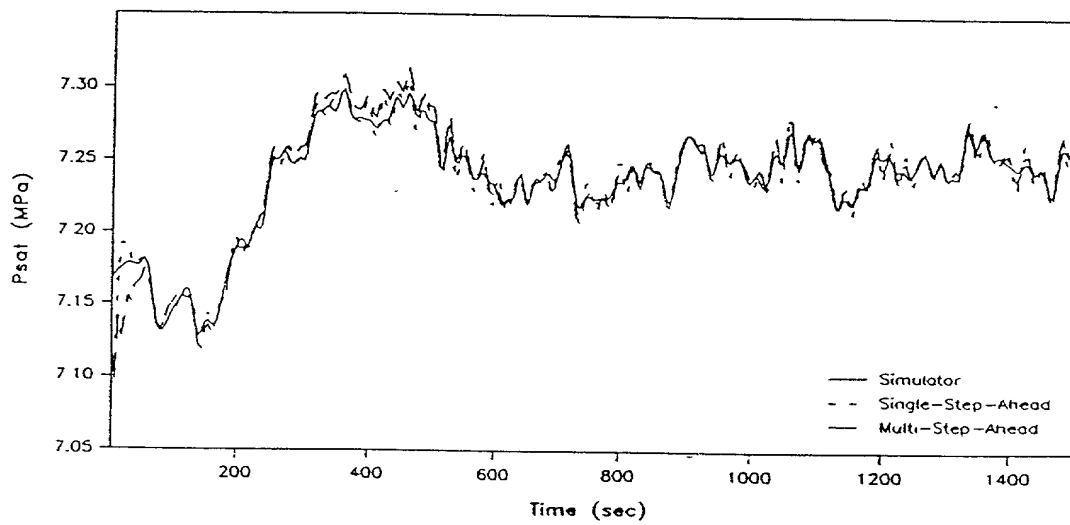
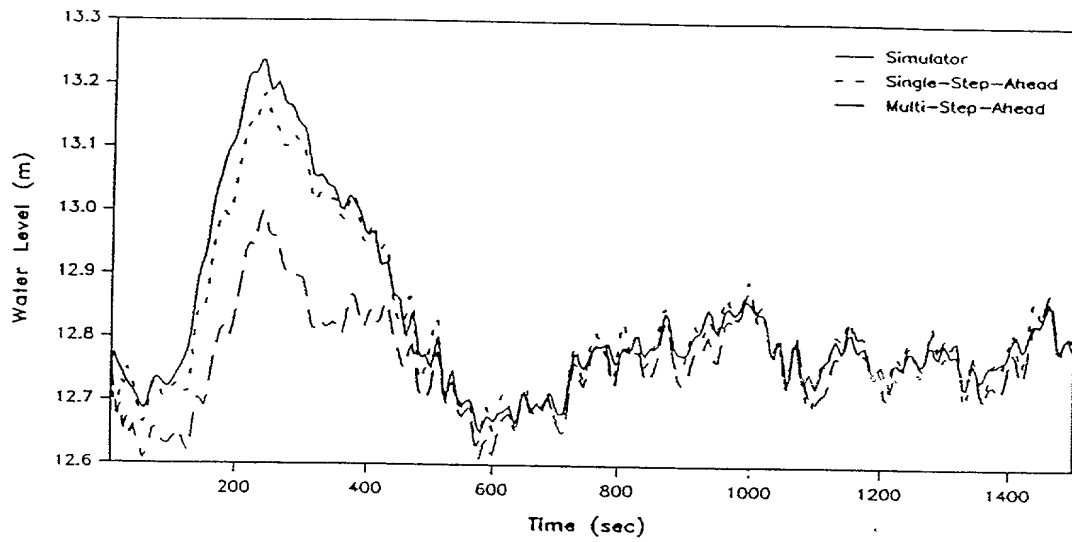


Figure 62. UTSG Transient Response Prediction for a 45% to 35% of Full Power Ramp.

231

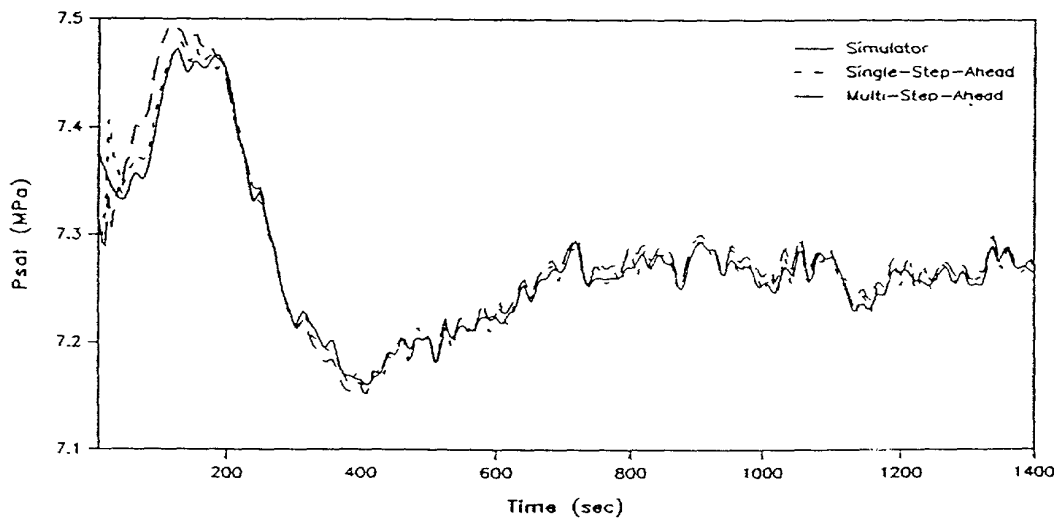
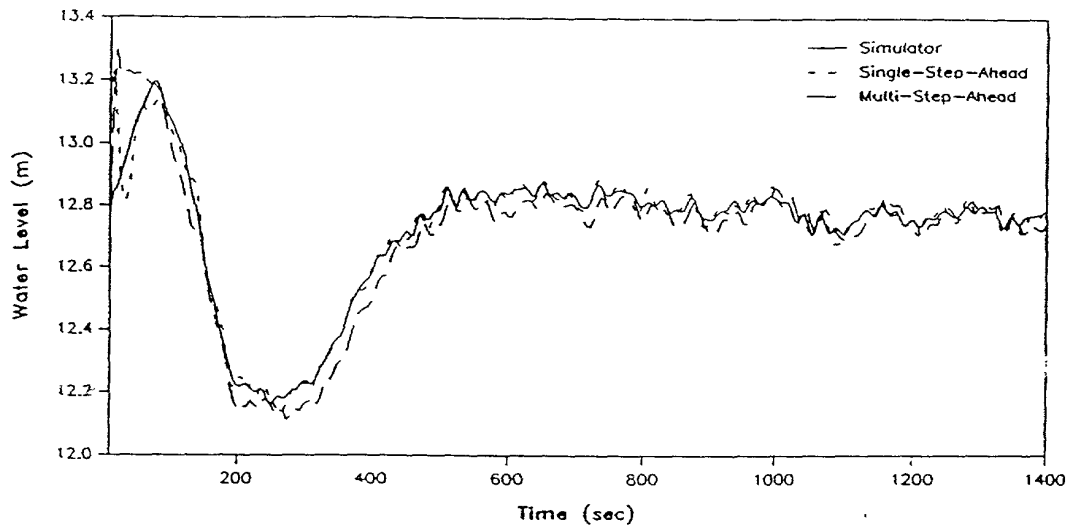


Figure 63. UTSG Transient Response Prediction for a 20% to 35% of Full Power Ramp.

232

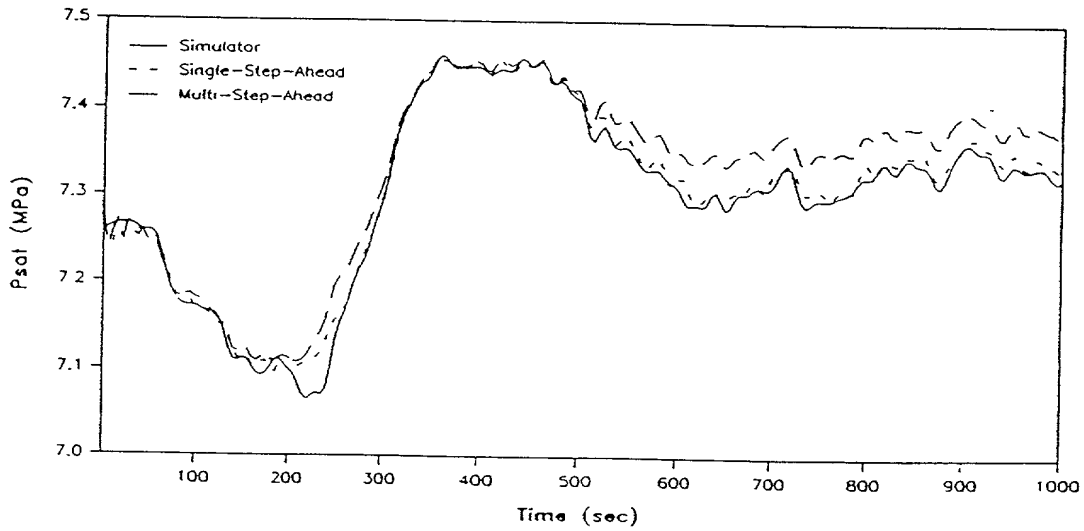
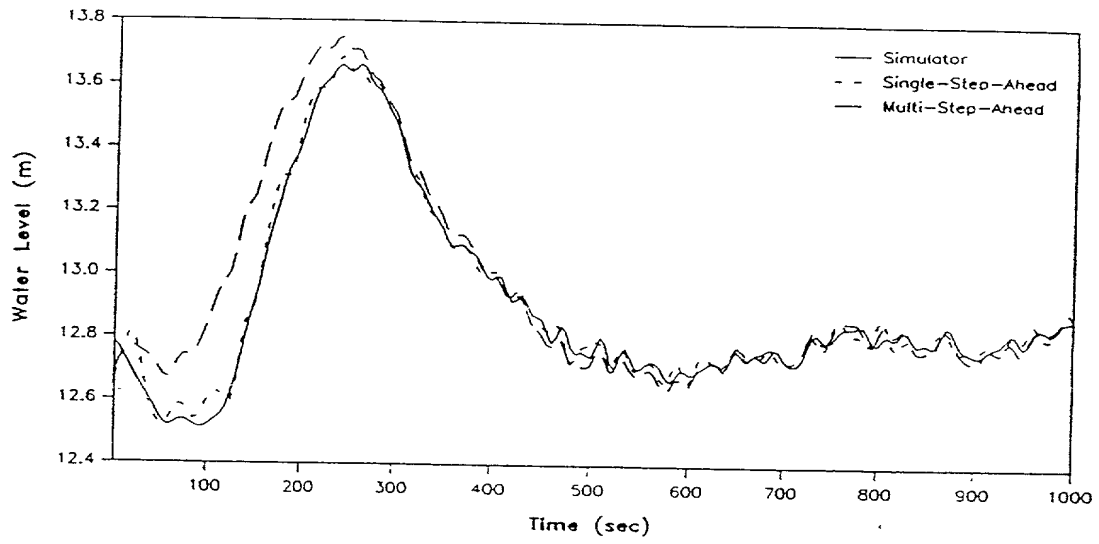


Figure 64. UTSG Transient Response Prediction for a 35% to 20% of Full Power Ramp.

233

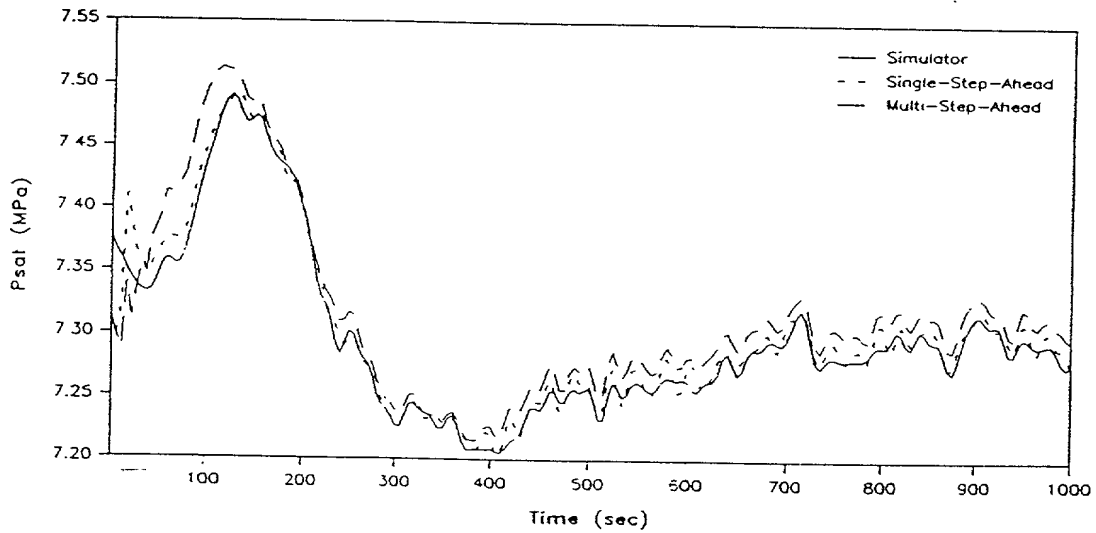
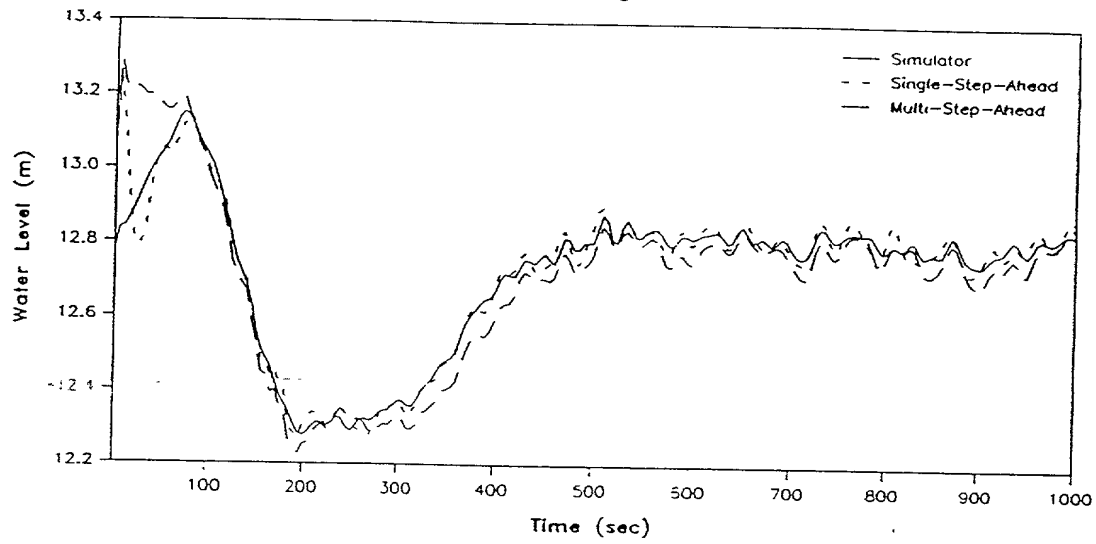


Figure 65. UTSG Transient Response Prediction for a 20% to 30% of Full Power Ramp.

234

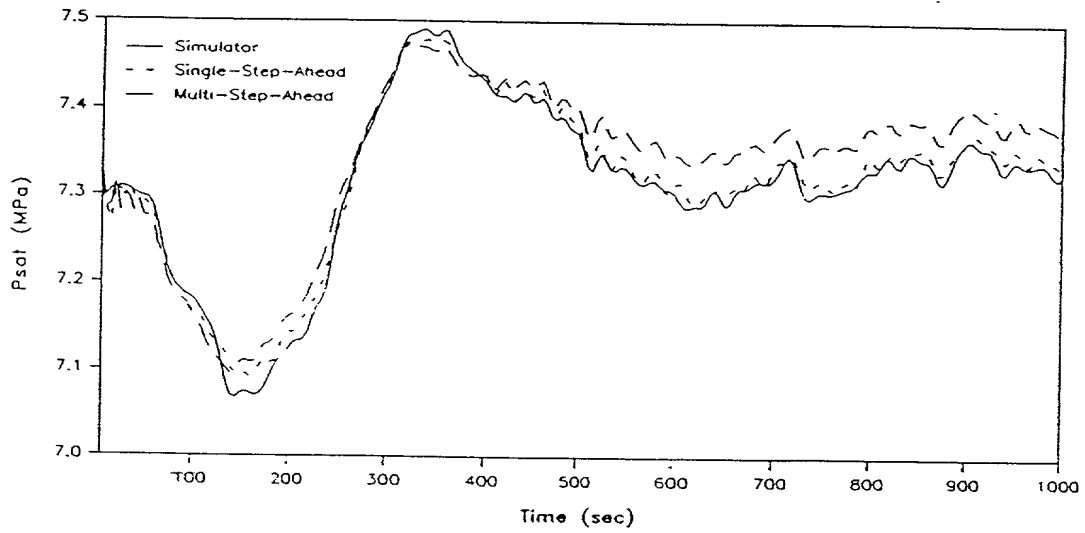
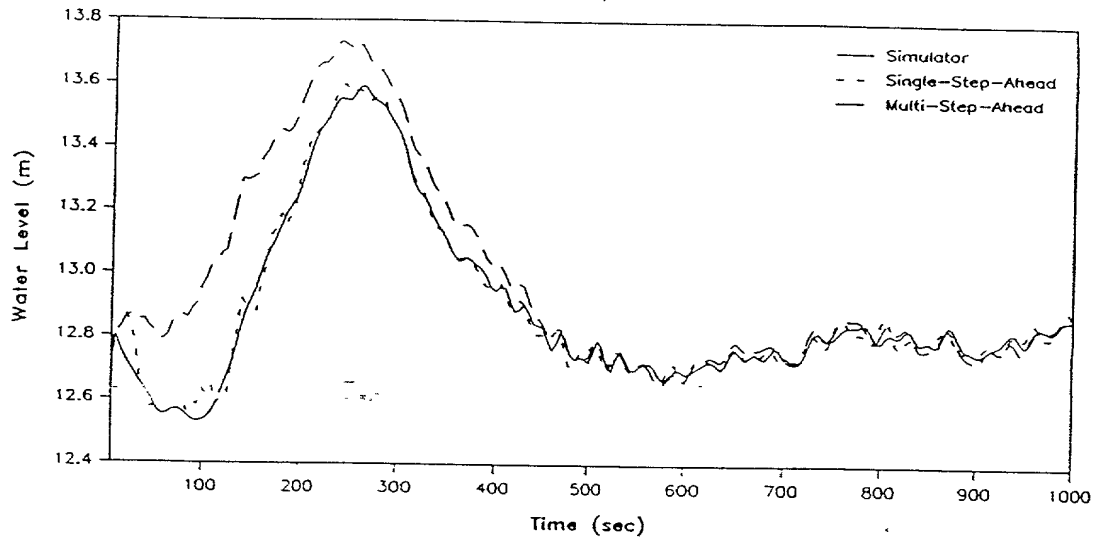


Figure 66. UTSG Transient Response Prediction for a 30% to 20% of Full Power Ramp

Table 6. UTSG Empirical Model Validation at High Power Levels.

Test Case	SSP	MSP
Water Level	RMSE [Max R.Error]%	RMSE [Max R.Error]%
50% to 55% Step	0.076 [0.25]	0.59 [0.15]
60% to 65% Step	0.097 [0.39]	0.31 [0.13]
80% to 85% Step	0.170 [0.84]	0.28 [0.14]
90% to 95% Ramp	0.200 [0.94]	0.59 [0.23]
50% to 95% Ramp	0.085 [0.54]	0.30 [0.16]
95% to 50% Ramp	0.018 [1.54]	0.70 [0.26]
60% to 80% Ramp	0.091 [0.57]	0.72 [0.13]
80% to 60% Ramp	0.130 [1.13]	0.57 [0.20]
50% to 75% Ramp	0.077 [0.54]	0.96 [0.11]
75% to 50% Ramp	0.120 [1.13]	0.63 [0.18]
Saturation Pressure	RMSE [Max R.Error]%	RMSE [Max R.Error]%
50% to 55% Step	0.099 [0.54]	0.13 [0.52]
60% to 65% Step	0.170 [1.10]	0.20 [1.23]
80% to 85% Step	0.079 [0.41]	0.11 [0.68]
90% to 95% Ramp	0.057 [0.15]	0.09 [0.26]
50% to 95% Ramp	0.110 [0.66]	0.21 [1.28]
95% to 50% Ramp	0.096 [0.60]	0.15 [0.85]
60% to 80% Ramp	0.120 [1.22]	0.18 [1.40]
80% to 60% Ramp	0.054 [0.16]	0.08 [0.47]
50% to 75% Ramp	0.220 [2.10]	0.30 [1.80]
75% to 50% Ramp	0.082 [0.27]	0.11 [0.54]

Figure 78 shows the water level response to a steam flow rate drop from 100% of full power level to 5% of full power level. As expected the water level increases very quickly in this open-loop simulation, where no control action is provided. In this figure both the MSP and the SSP water level predictions are grossly inaccurate. Figure 79 shows the water level response to feedwater flow rate drop from 100% of full power level to 5% of full power level. As expected, the water level decreases very quickly in this open-loop simulation, where again no action control is provided. Also, in this figure

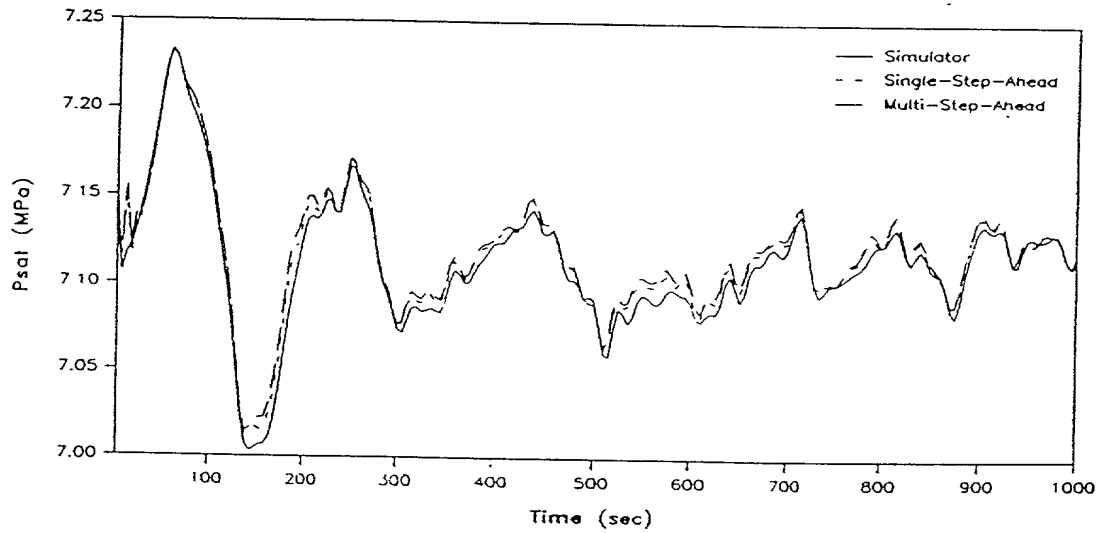
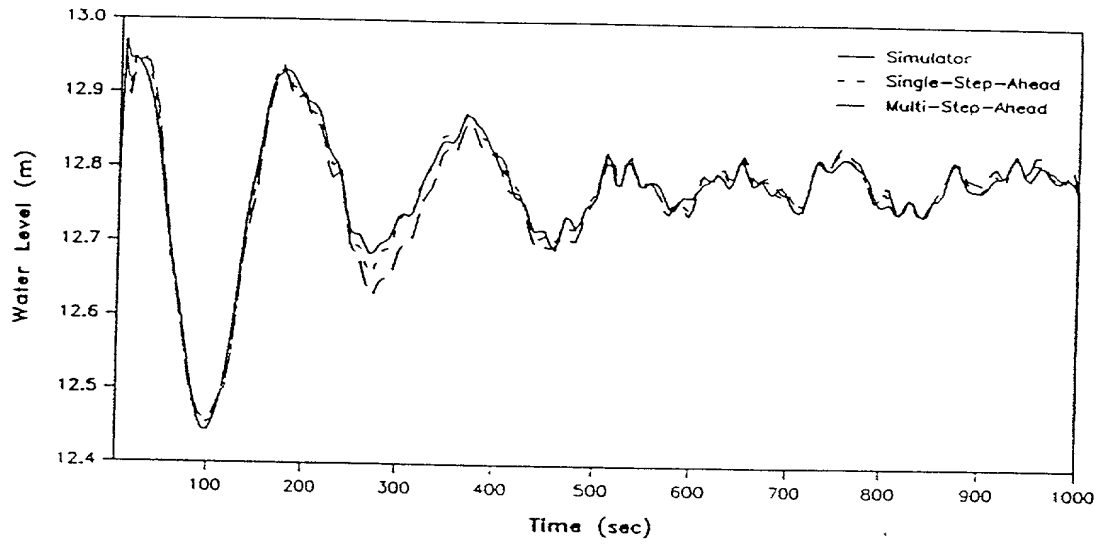


Figure 67. UTSG Transient Response Prediction for a 50% to 55% of Full Power Step.

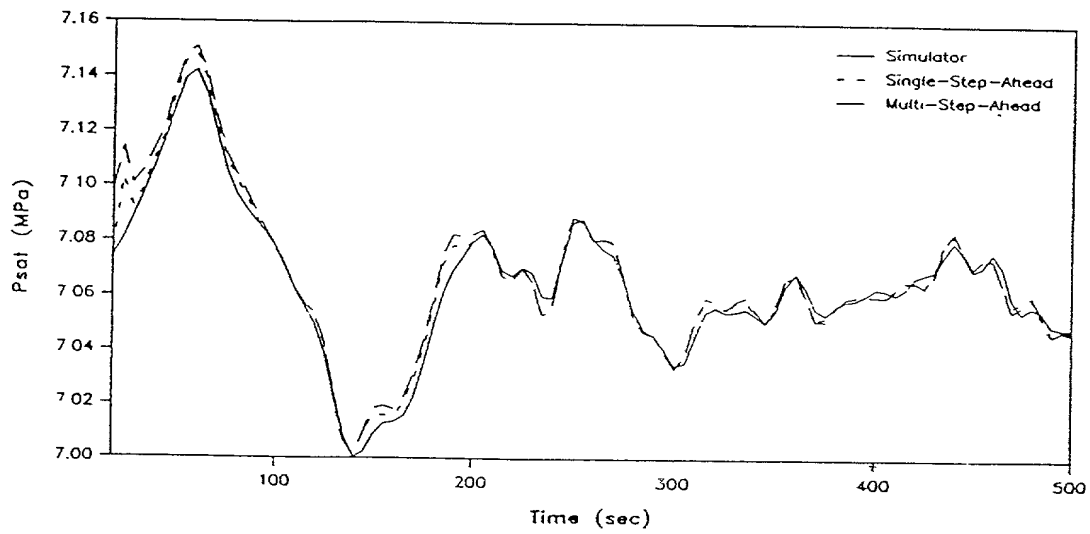
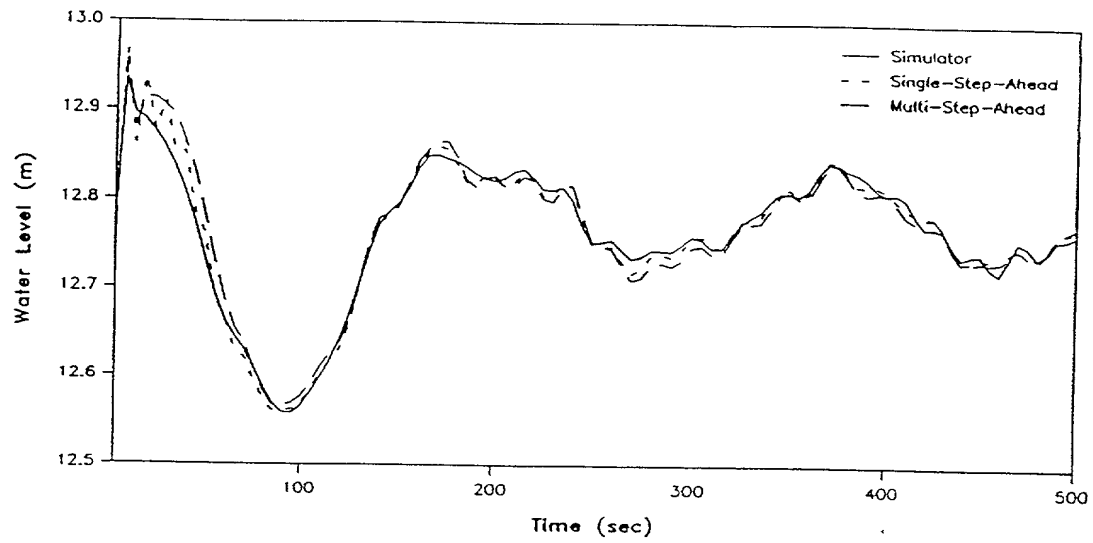


Figure 68. UTSG Transient Response Prediction for a 60% to 65% of Full Power Step.

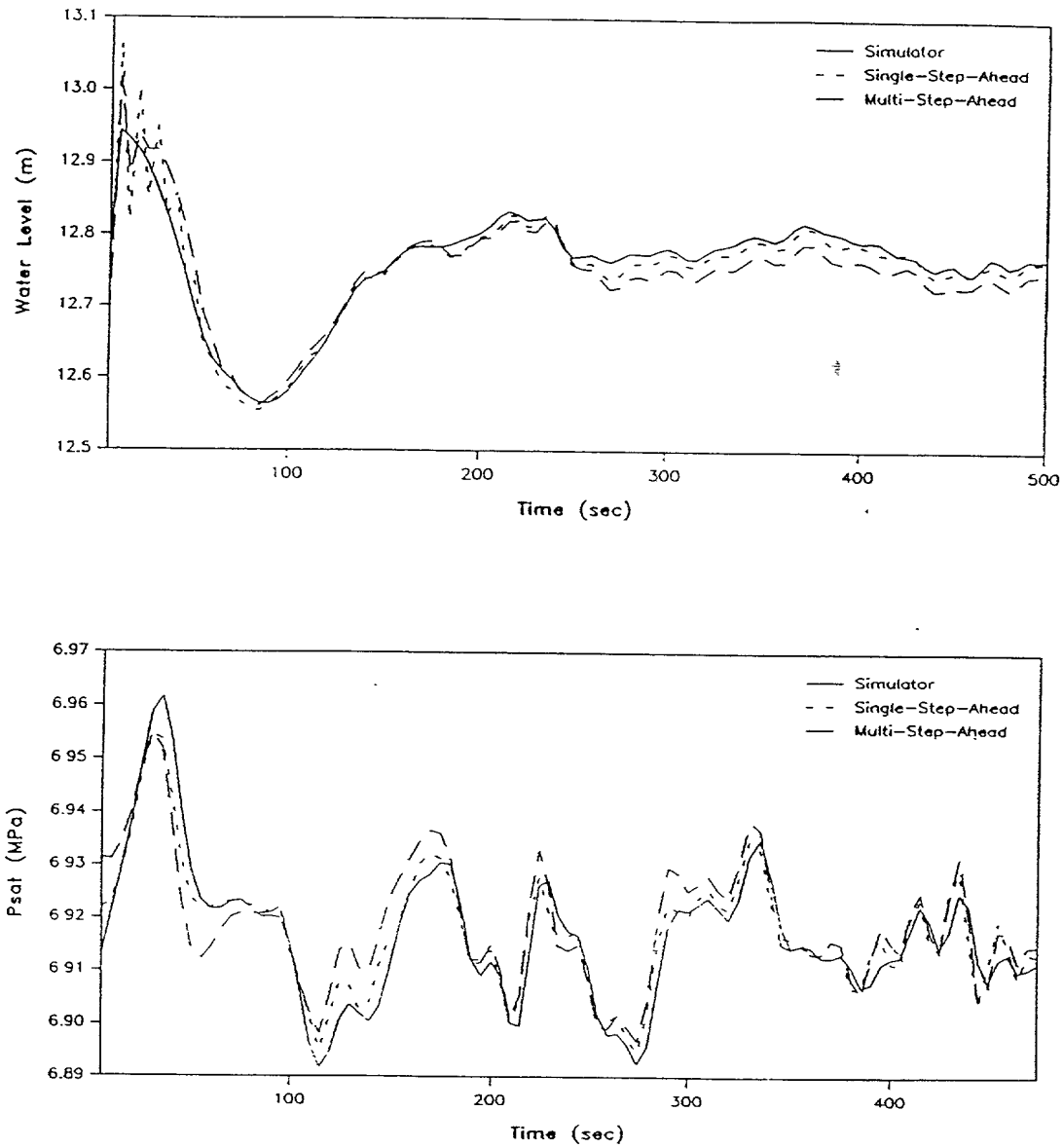


Figure 69. UTSG Transient Response Prediction for a 80% to 85% of Full Power Ramp.

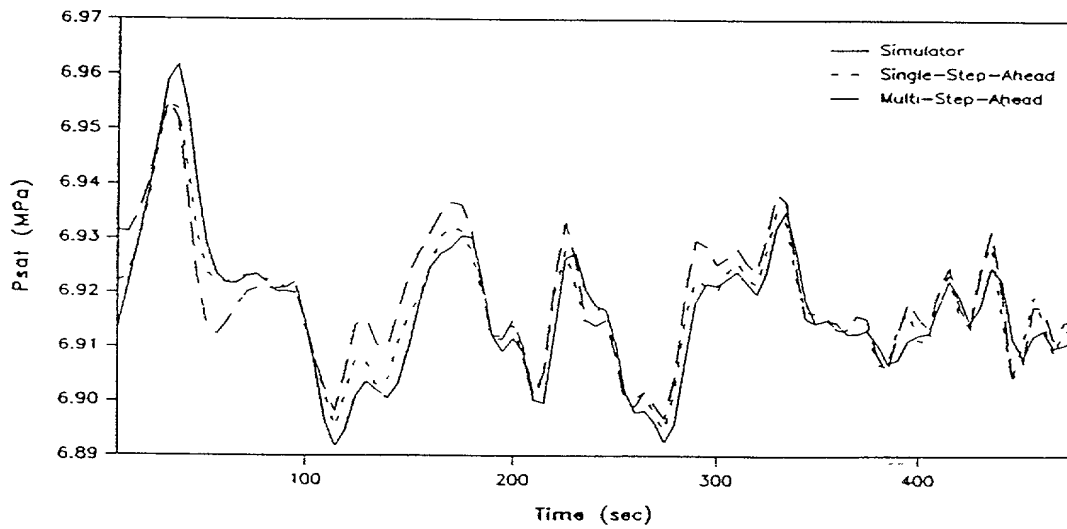
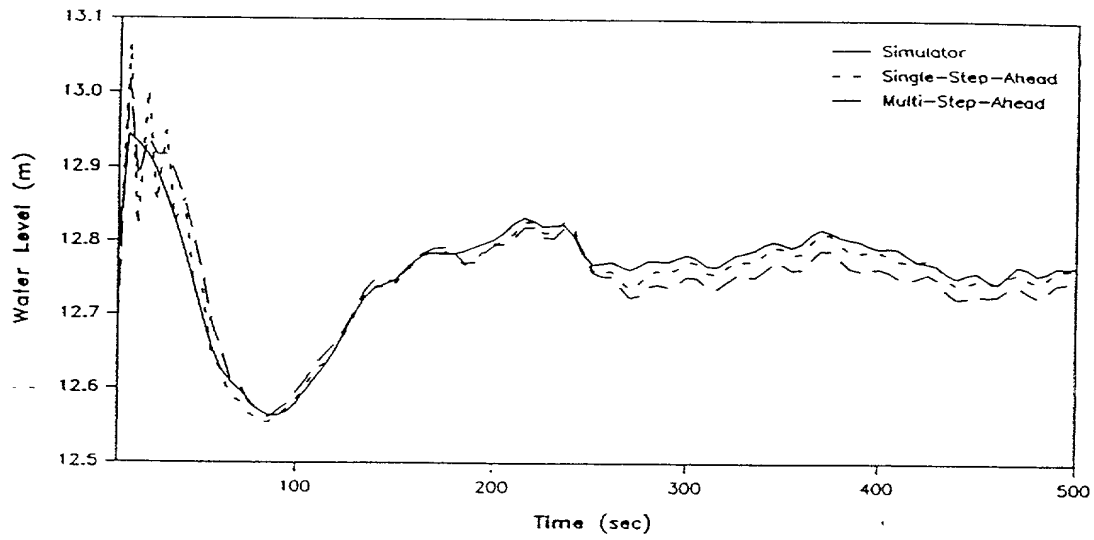


Figure 70. UTSG Transient Response Prediction for a 90% to 95% of Full Power Ramp.

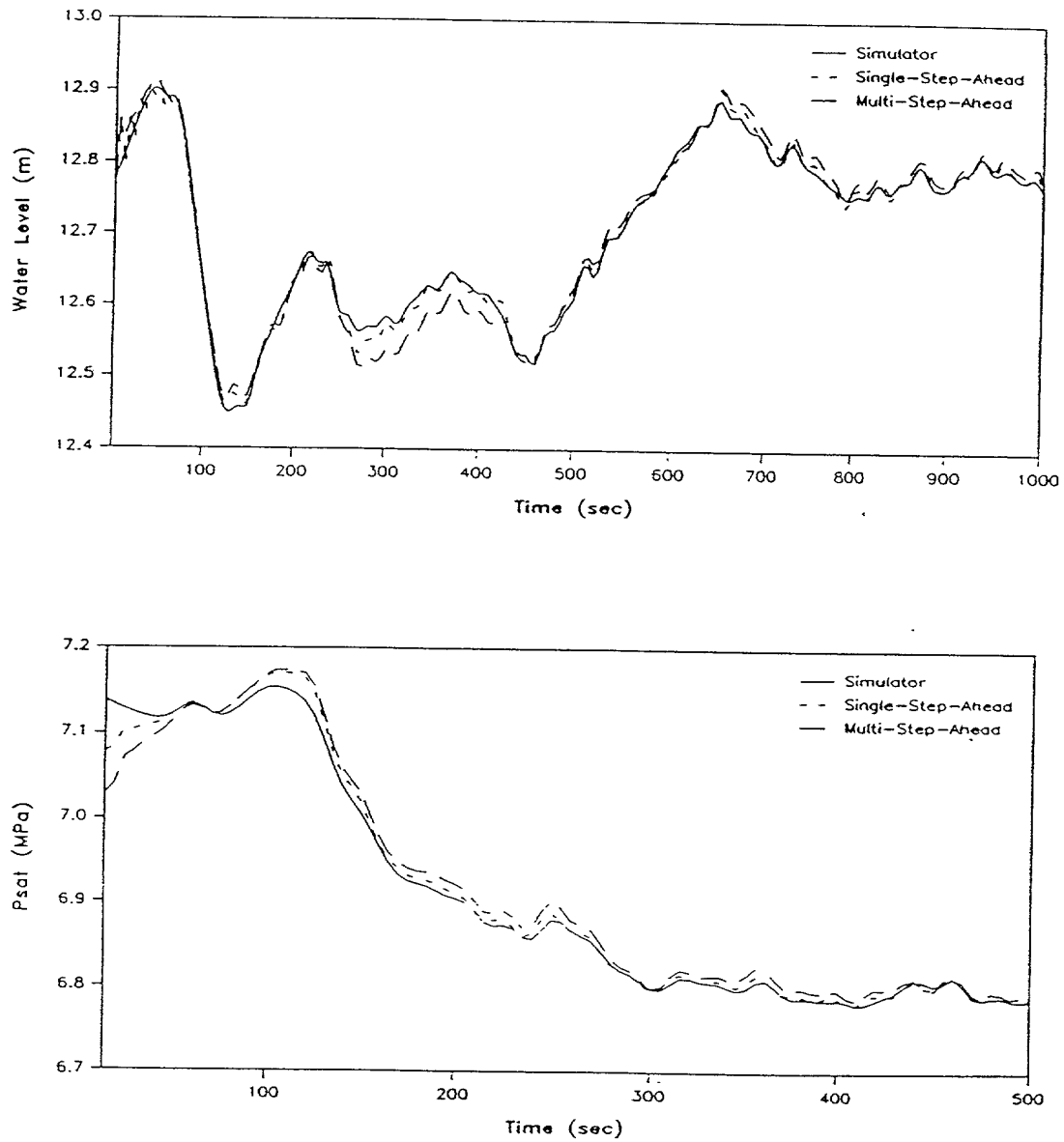


Figure 71. UTSG Transient Response Prediction for a 50% to 95% of Full Power Ramp.

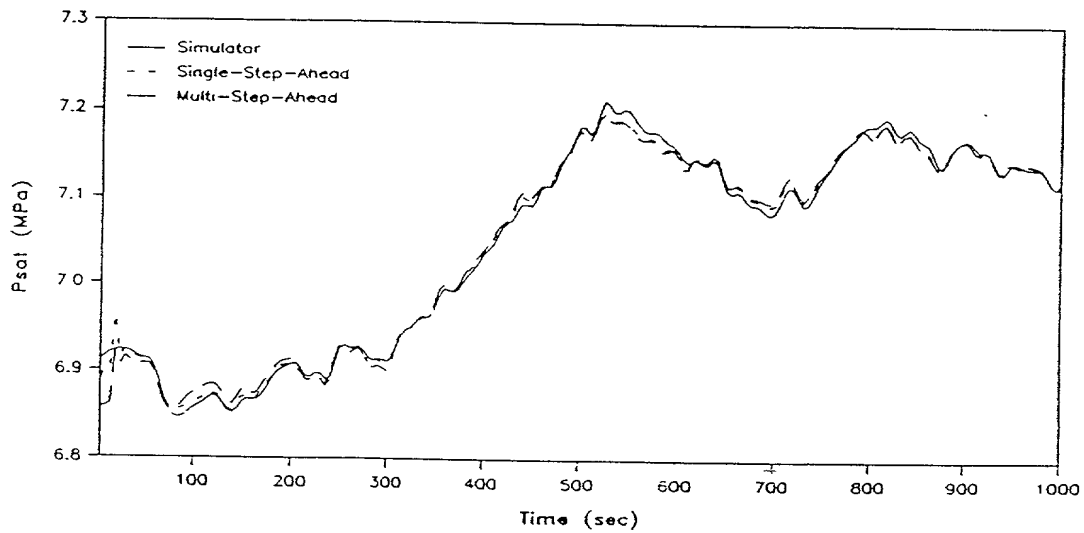
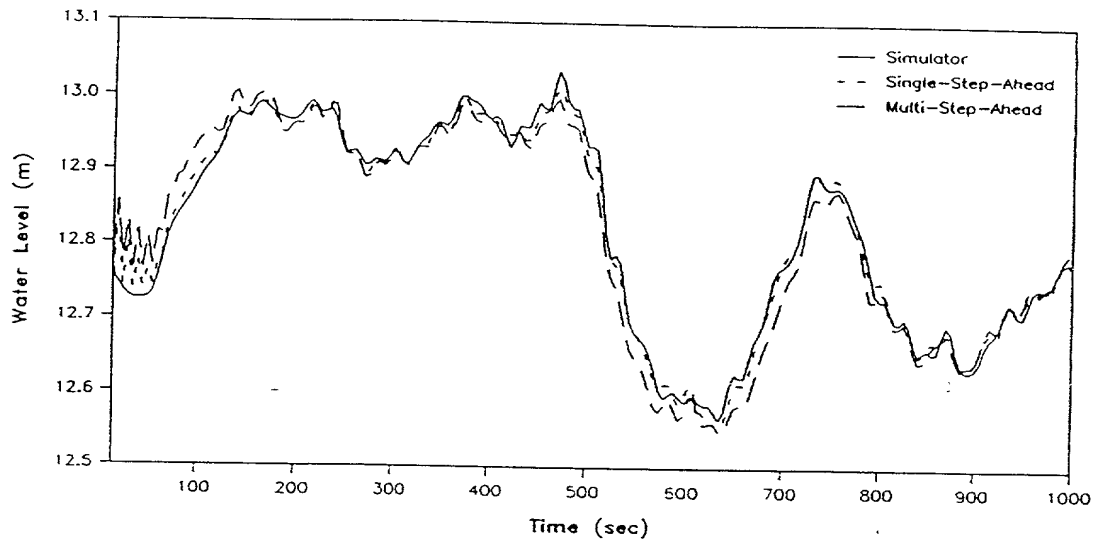


Figure 72. UTSG Transient Response Prediction for a 95% to 50% of Full Power Ramp.

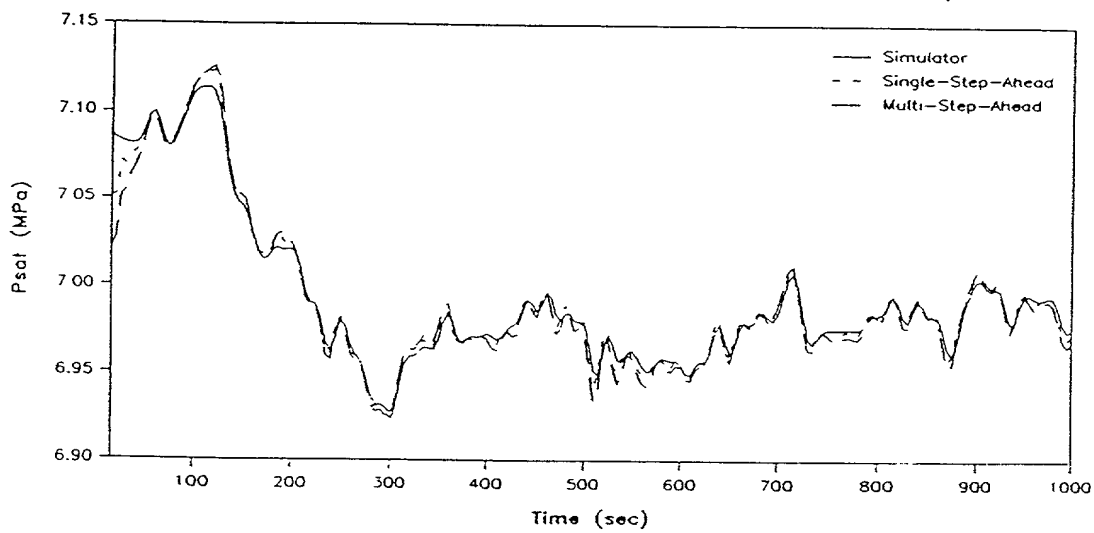
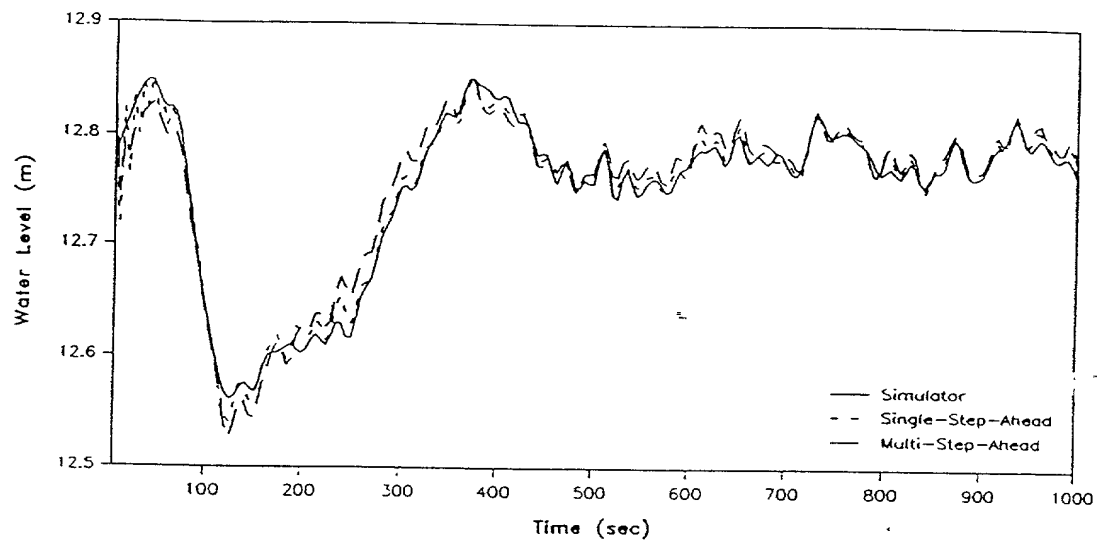


Figure 73. UTSG Transient Response Prediction for a 60% to 80% of Full Power Ramp.

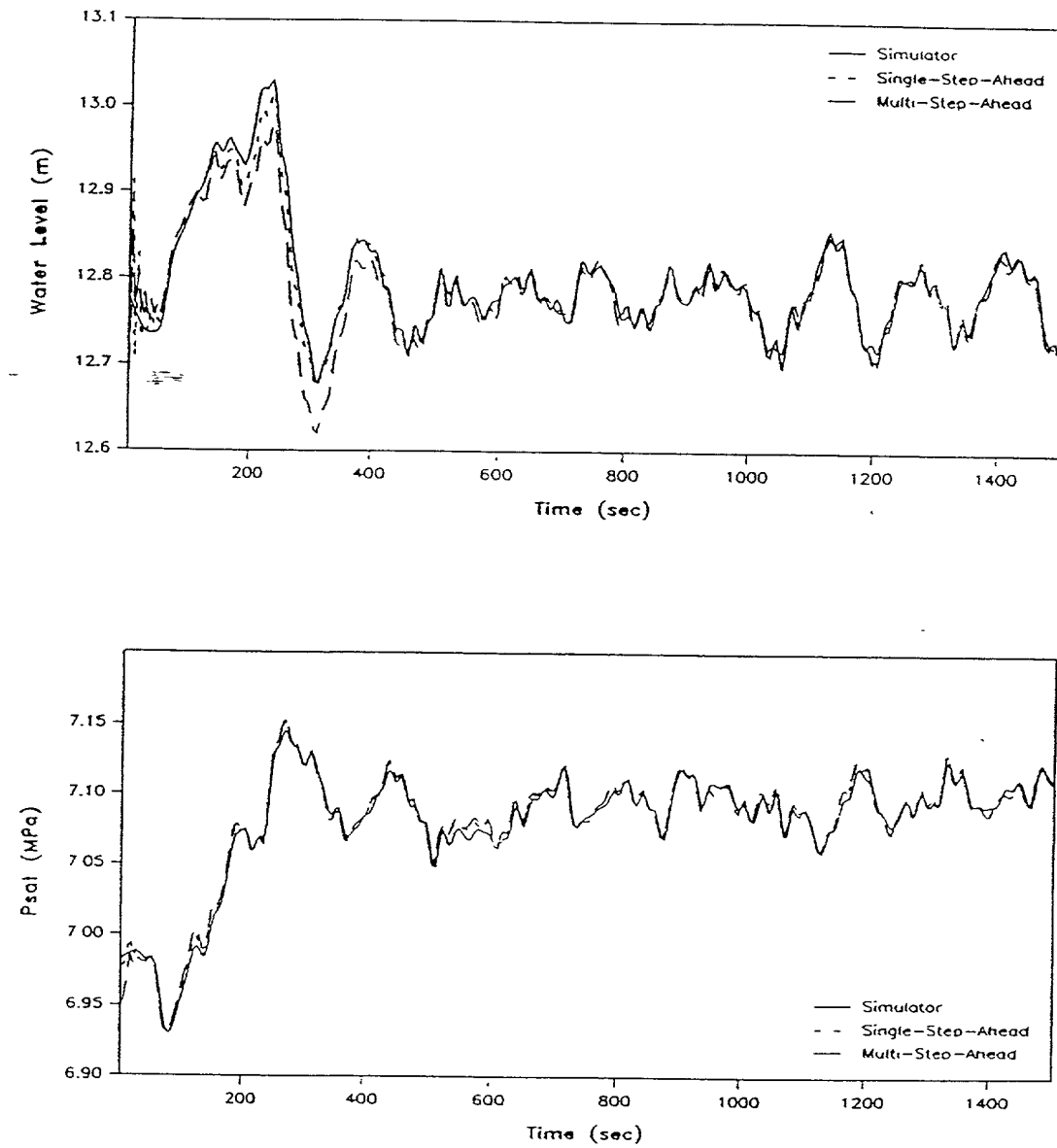


Figure 74. UTSG Transient Response Prediction for a 80% to 60% of Full Power Ramp.

244

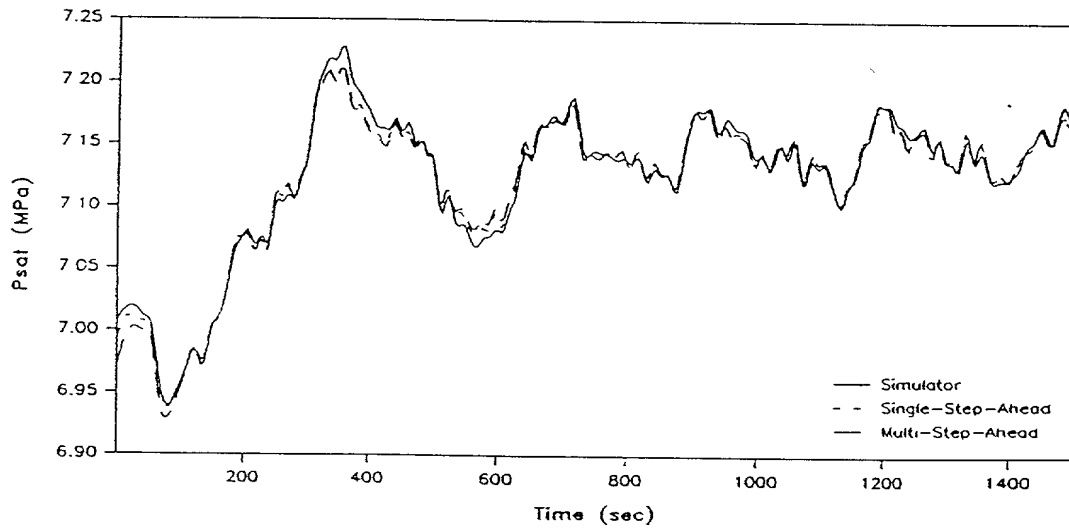
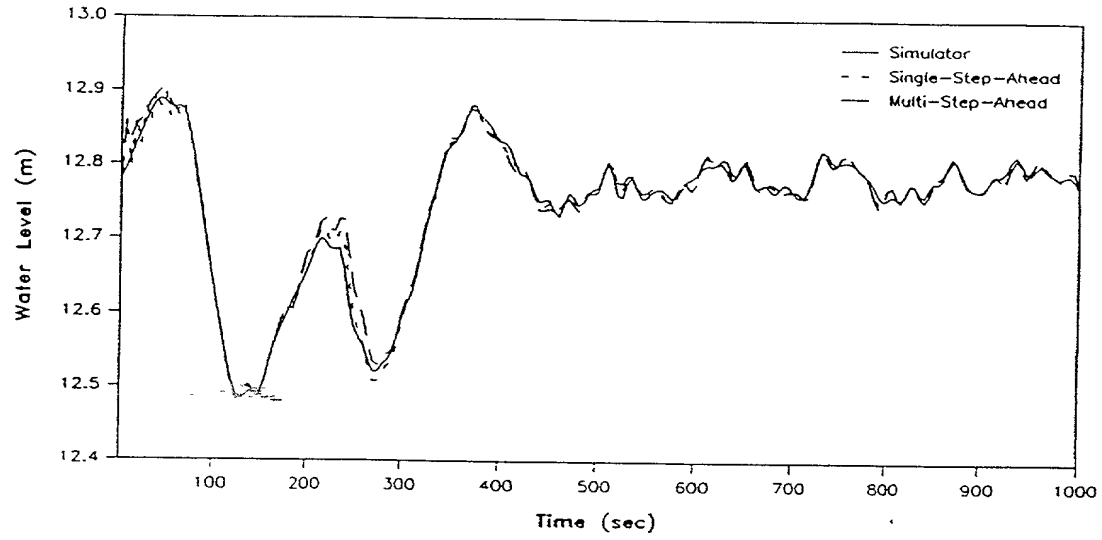


Figure 75. UTSG Transient Response Prediction for a 50% to 75% of Full Power Ramp.

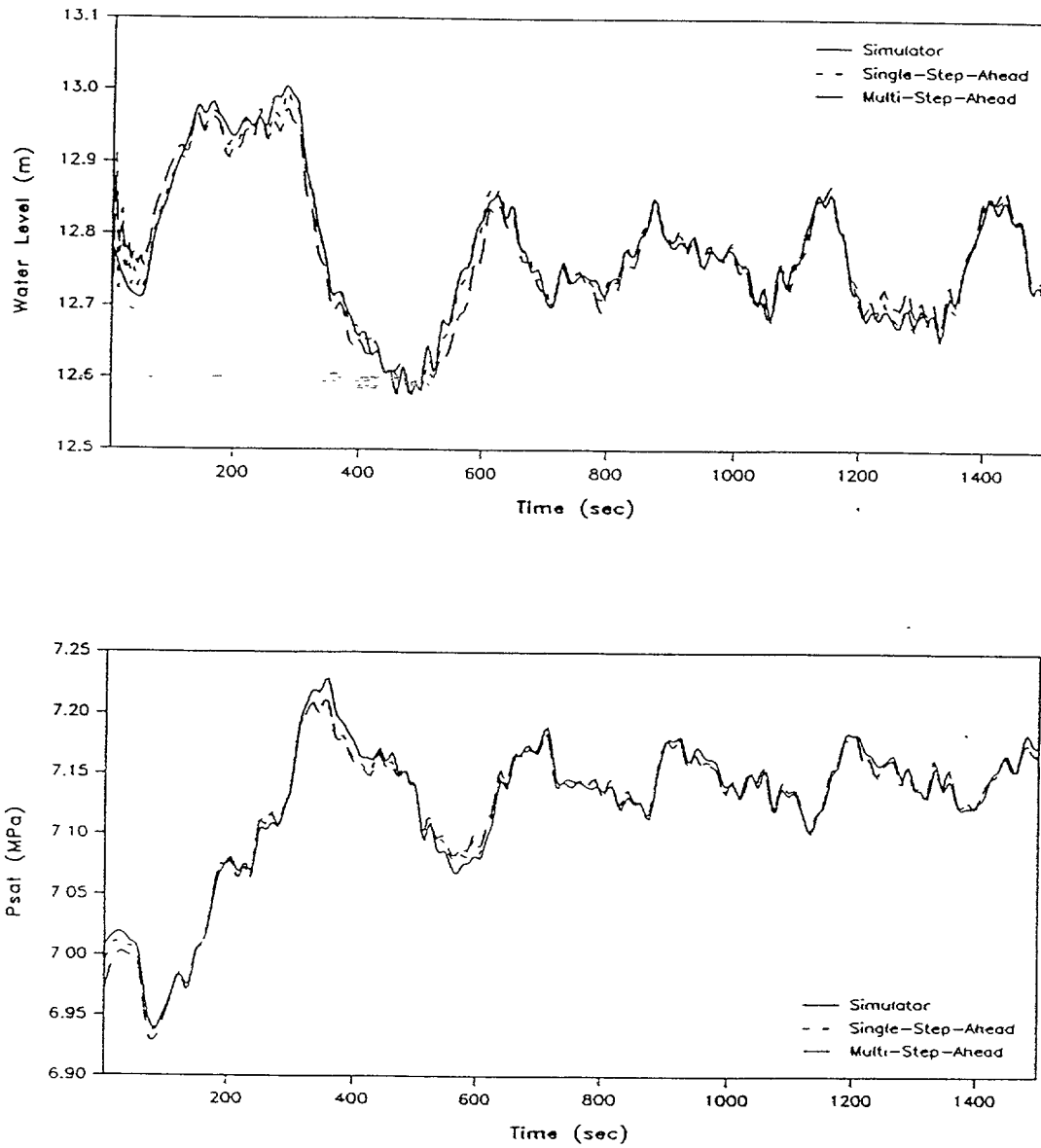


Figure 76. UTSG Transient Response Prediction for a 75% to 50% of Full Power Ramp.

both the MSP and the SSP water level predictions are grossly inaccurate. These tests indicate that the designed CNN predictors have certain performance limitations.

V.6 Chapter Summary

Nonlinear SI of a UTSG using an RMLP network with TF and GF is performed. A conventional nonlinear SI (polynomial NARX) method is also used. This latter totally failed to give a predictor capable of performing MSP, while the TF CNN partially failed to give a CNN predictor capable of MSP. The GF CNN method succeeded in giving a predictor capable of MSP. Six CNN predictors were designed using this method to cover the entire operating range of the UTSG. Extensive validation tests were performed to validate all the CNN predictors obtained. The performance of GF CNN predictors in the medium and high power ranges was better than the one in the low power range. This is because of the high level of process and sensor noise at low power levels, and also because of the nonminimum phase effects which are more pronounced at the low power levels. A draw-back of the GF CNN method is the need for proper initialization of the weights and biases, as well the need for appropriate choice of the number of hidden layers and hidden nodes required to construct the neural network. However, the guidelines given in this research study provide a good set of procedures for obtaining the appropriate network architecture when modeling complex process systems.

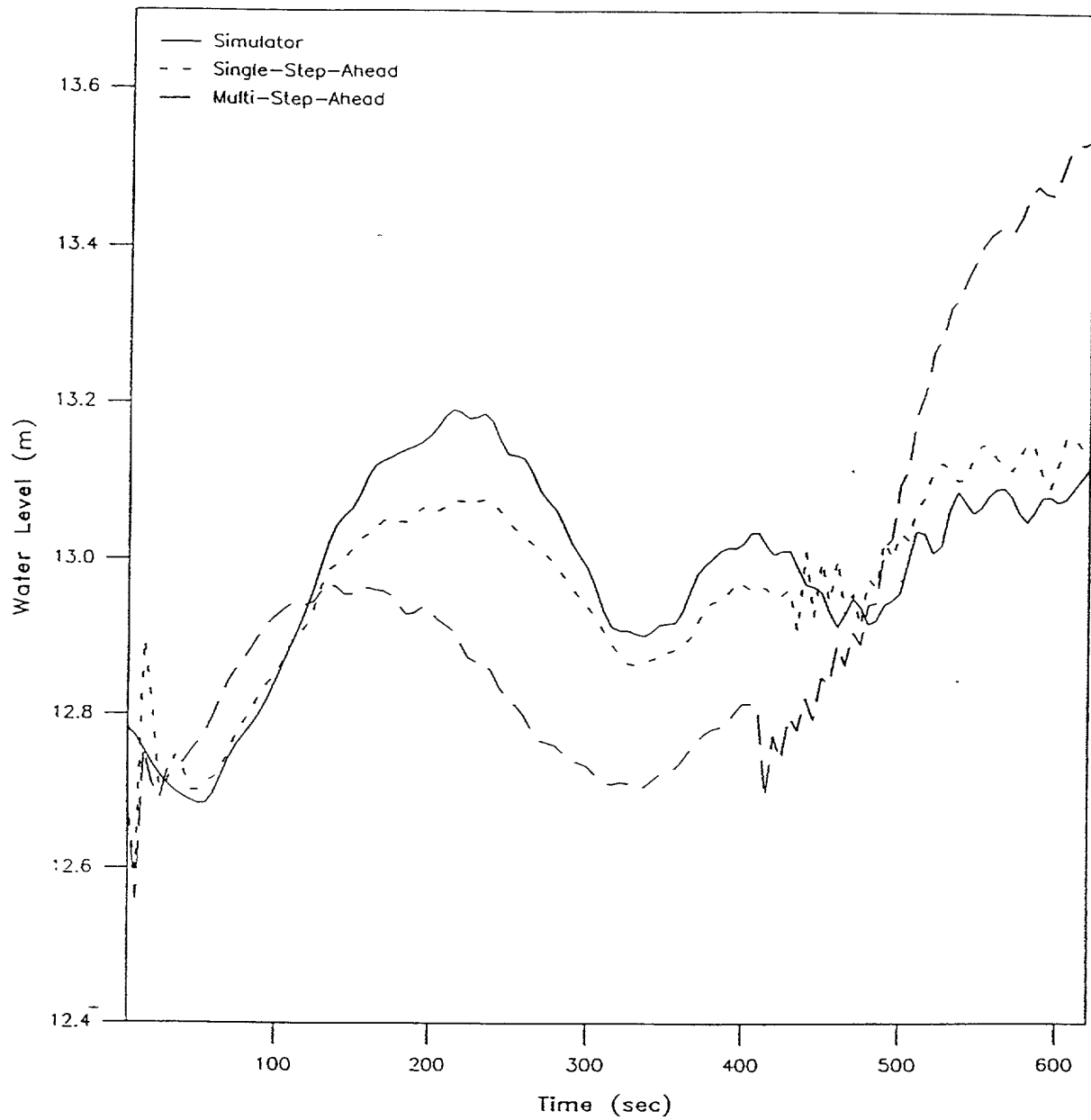


Figure 77. UTSG Water Level Closed Loop Response Prediction to a Lost of Load from 100% to 5% of Full Power.

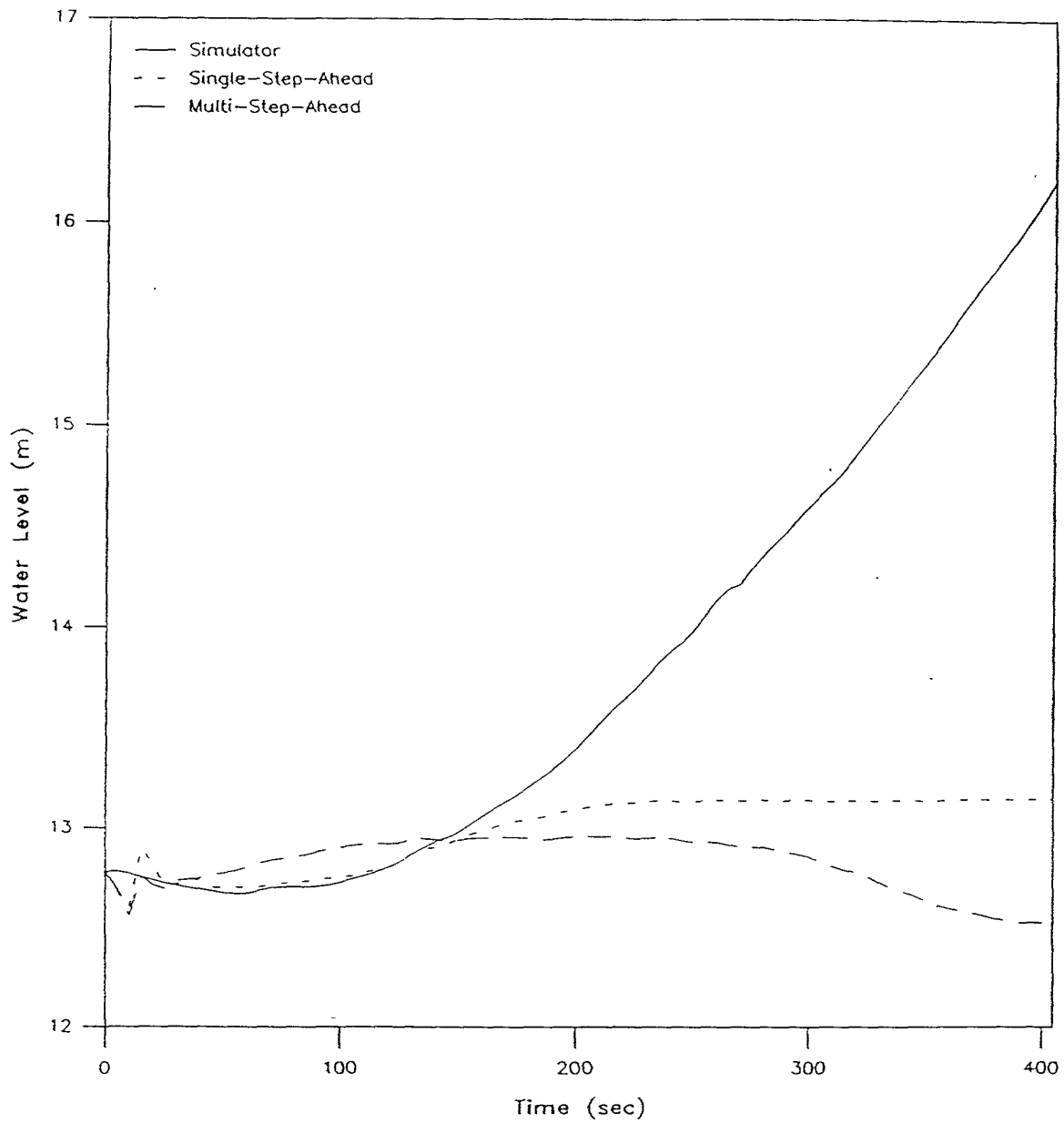


Figure 78. UTSG Water Level Open Loop Response Prediction to a Steam Flow rate drop from 100% to 5% of Full Power.

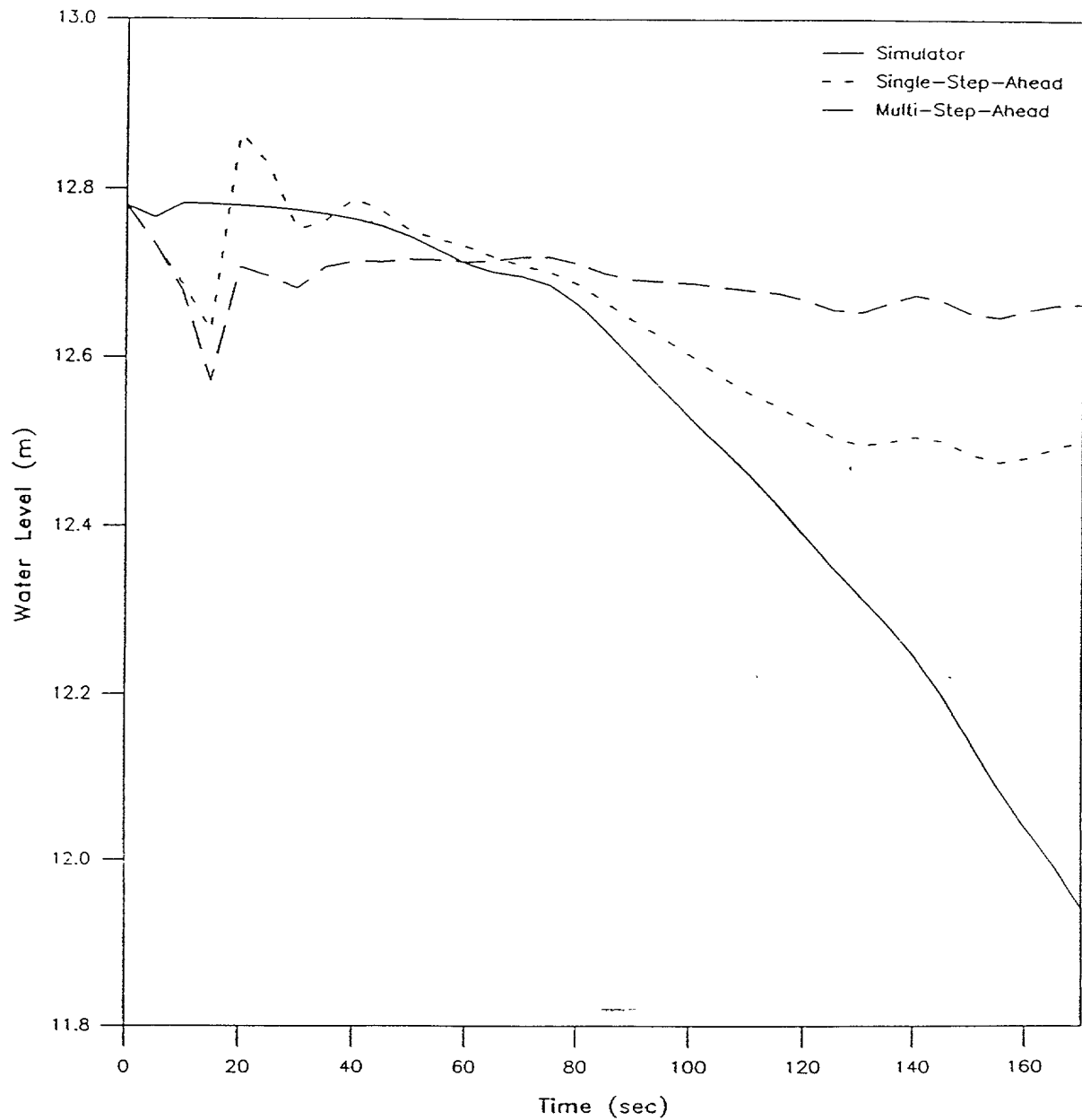


Figure 79. UTSG Water Level Open Loop Response Prediction to a Feed Water Flow rate drop from 100% to 5% of Full Power.

SUMMARY AND CONCLUSIONS

VI.1 Summary

The objective of this research study was to develop a procedure for designing predictors for complex process systems, capable of performing accurate multi-step-ahead prediction (MSP) and single-step-ahead prediction (SSP). The proposed procedure was tested on an artificial system, as well as on a real-world complex process system, namely a U-Tube Steam Generator (UTSG). A model with the aforementioned capabilities has many applications in the areas of forecasting and prediction, control engineering, and condition monitoring and fault diagnosis. Use of conventional methods for modeling, utilizing first principles, has many disadvantages, such as development and execution cost, complexity, potential inaccuracies to drifts, and inability to incorporate aging, and wear and tear effects. Recently, an emerging approach has been to use certain classes of CNNs as model structures for nonlinear System Identification (SI). This has been motivated by the well-known characteristics of CNNs for approximating static nonlinear mappings. Thus, in this research perceptron-based CNNs are used as model structures for nonlinear SI.

In chapter II, a general description of SI was given. First, the intended purpose and objectives of SI must be specified before selecting and applying the appropriate procedures. This is followed by experiment design, in order to collect the data needed for the empirical modeling step. It is up to the user to determine which variables to

measure, and how often to measure them. The input signals must satisfy certain conditions, and must be sufficiently rich during the experiment so that to excite all of the dynamic modes of interest. Apriori knowledge about the system under study is a key factor in experiment design and data collection, as well as all other SI stages. The next step in SI is model structure selection. This is the most important stage in SI. At this stage engineering intuition, and insight must to be combined with formal properties of the model. Model parameter estimation is the next stage in SI. In this stage the parameters of the model structure chosen are estimated using algorithms, such as the least-squares method. The final stage in SI is model validation. The purpose of this stage is to test whether or not the designed model is valid for the intended purpose. An effective way to validate a model is to test it using new data not used during its design process.

Empirical modeling using CNN techniques has many applications. A detailed discussion of these applications was also given in chapter II. Forecasting and prediction is one such area. Forecasting future events is performed by extrapolating the models beyond the range over which they were estimated. The term forecasting is often thought of as applicable solely to problems in which the future event is predicted. Nevertheless, information provided by the forecasting process can be used in several ways. Forecasts are also useful as guides for model building. A forecast which is found to be off the target when the actual observations are available, provides information which might lead to the revision of the model that provided the forecast.

Another application of CNN models is in control engineering, namely in neurocontrol. There are five basic designs used in the area of neurocontrol. Namely, supervised

control, direct inverse control, neural-adaptive control, backpropagation of utility, and the adaptive critic methods [66]. All of these methods were discussed in details in chapter II as well.

A third application which is gaining a lot of industrial interest is the area of condition monitoring and fault diagnosis. Fault-tolerance in dynamic systems is traditionally achieved through the use of hardware redundancy, where protection against localized damage is provided by spatially distributing repeated hardware elements around the system. However, hardware redundancy comes with extra cost, software, and additional space to accommodate the equipment. New approaches have been developed which seek to eliminate all of the redundant hardware. These new approaches are based on the idea that two (or more) dissimilar sensors measuring different variables can be used in a comparison scheme to detect any fault in the system. The logic behind this idea is that although the sensors are dissimilar in terms of the signals each sensor is measuring, they are all driven by the same dynamic states of the system and they are therefore functionally related [48]. These functionally redundant schemes are basically signal processing techniques employing state estimation, parameter estimation, and statistical decision theory, all of which can be performed using software. Fault detection schemes are therefore designed under the assumption that either the dynamic states of the system under study are known to a certain precision, or it is possible to determine the values of certain physical parameters by on-line identification techniques. CNNs appear to be the key for the success of these fault diagnosis schemes.

Having stated the potential and critical necessity of CNNs in the aforementioned application areas, the objective was then to investigate in detail some classes of CNNs. Chapter III is a comprehensive overview of a certain class of CNNs. Namely, well-known the Feed-forward Multilayer Perceptron (FMLP) neural network and the more recent Recurrent Multilayer Perceptron (RMLP) neural network are discussed. The RMLP network is developed for use in nonlinear SI. It has a feedforward part that performs nonlinear curve-fitting, and one-time-delayed cross-talk and local recurrency connections in the hidden layers which serve as local memory for capturing temporal correlation. One of the training algorithms developed for training the RMLP network is the dynamic gradient descent learning algorithm, which is a parameter estimation technique adopting , as its name implies, a gradient descent method. The basic mechanism behind this learning rule is the adjustment of the network weights and the bias terms, until the Mean-Squared Error (MSE) between the output predicted by the network and the target output, the sensor reading in this case satisfies a certain stopping criterion.

Two learning algorithms based on the dynamic gradient descent were considered, the Teacher Forcing (TF) algorithm and the Global Feedback (GF) algorithm. TF means that the network input layer includes the latest sensed output(s) of the system. In other words, at each time step the network input layer is updated using the latest sensed output(s). GF on the other hand, means that the network input layer includes the latest output(s) predicted by the CNN itself. The fact that the predicted output(s) are utilized by the network in GF, as compared to the sensed output(s) in TF, makes the derivation of the gradients in GF more complex than in TF. Moreover, in GF

the network learns the input-output data based on its own MSP only. This feature enables the network to perform well in MSP on the training set. By approximately designing such a MSP predictor, good performance can be obtained on other data sets not seen during training.

Deriving the appropriate learning algorithms is a necessary but not a sufficient condition for accurate MSP. The other condition is to design the appropriate network architecture. Network architecture design is the task of assigning the appropriate number of hidden layers and hidden nodes of the neural network. Utilizing the appropriate delayed input(s) and/or output(s) in the network input layer is also a critical issue in designing the architecture of the network. The issue in successful network architecture design is to prevent the CNN from overfitting. By overfitting, the network's output fits the training data too closely. The form of the mapping function used by CNNs is so flexible and the gradient descent learning process is so relentless that, unless it is prevented, overfitting is a serious hazard.

The solution to overfitting is therefore to limit the network computational power. This can be done by allowing the network enough degree of freedom to provide a good fit on the training set, but not so many degrees of freedom that it risks overfitting. Network complexity can be limited by four means: choosing the suitable number of hidden layers, limiting the number of hidden nodes, utilizing the appropriate number of delayed inputs and/or past outputs, and choosing the correct stopping criterion during learning. All of these means for controlling network complexity were addressed in detail in chapter III.

Part of the investigation of CNNs was to examine all the aforementioned algorithms and techniques using a simple case-study. This was the objective of chapter IV, where a case-study was presented comparing the conventional methods of SI and the CNN-based methods. The objective was to build a model capable of MSP. The case-study was a two input two output nonlinear system with three dynamic states. The training data consisted of a combinations of steps and pulses of different magnitudes mixed with some white noise.

The polynomial NARX model structure used in this SI study was estimated using the forward-regression orthogonal method. A detailed presentation of this technique was given in chapter IV. Many models were obtained using this technique, however, only the model which gave the best performance was presented. Three types of tests were performed with signals unknown during identification, for investigating the models' predictive performance. All these tests were performed with both low and high noise levels. The polynomial NARX model gave satisfactory results in both SSP and MSP, and both in the presence of low and high noise levels.

In utilizing CNN for SI, many options were examined: i.e. utilizing delayed input(s) and/or delayed output(s) in the input layer, batch weight update vs. individual weight update during training, teacher forcing vs. global feedback during training, weight decay, and making use of the model structure obtained using the polynomial NARX structure selection algorithm as the input layer of the neural network. Each of these options were discussed and presented in detail in chapter IV. The results of this investigation can be summarized as follows:

- (1) Individual weight update was found to perform better than batch weight update;

- (2) GF was found to outperform TF in MSP, but not in SSP, because the CNN-based predictor was designed to perform MSP as primary objective;
- (3) Utilizing more delayed output(s) and input(s) is as good as utilizing one delayed output. However, this finding is very much problem dependent and might not be applicable to other systems;
- (4) Weight decay did not yield any positive results;
- (5) Finally, the utilization of the structure detected by the polynomial NARX structure detection algorithm with CNNs did not improve the network's ability to perform MSP.

From the results obtained in this case-study using CNN-based methods and the polynomial NARX method, it was clearly demonstrated that CNN-based methods outperformed the best of the conventional method, the polynomial NARX, consistently. However, the improvements gained by using CNNs was not very significant, because the case-study is not a real-world problem, nor is it complex enough to observe major improvements. It is believed that the real power of CNN-based methods is to be demonstrated in a more complex system, such as the UTSG, and this was the objective of chapter V.

The UTSG is a complex process system with one control input, five disturbances, and three measured outputs. It consists of two fluid loops: the primary loop, and the secondary loop. A complete description of the UTSG was given in chapter V. In order to generate the data necessary for the empirical modeling using CNNs, an existing UTSG simulator was adopted for the purpose of this study. The simulator was developed using the one-dimensional mass, momentum, and energy conservation

equations. The simulator for this complex process system consists of nine nonlinear ordinary differential equations. Six differential equations for the secondary loop, and three for the primary loop. An integrated secondary-recirculation-loop momentum equation was incorporated into the simulator to calculate the water level. This simulator has been successfully validated against plant data for the whole power range of the UTSG. Because the open-loop system is unstable, a stabilizing feedback controller is required for system operation.

The empirical CNN model developed in this research study consisted of six separate predictors. Three models to predict the downcomer water level response at low power, medium power, and high power operation, respectively, and three other models to predict the secondary steam pressure response at low power, medium power, and high power operation, respectively. Using the simulator, a combination of step and ramp load level changes were generated at each power range to be used as training, testing, and validation data. Based on the results obtained from the case-study, it was clear that CNNs have better potential of identifying nonlinear systems. However, the polynomial NARX method was also investigated in this chapter. The objective was to compare the conventional method (polynomial NARX), and the CNN-based methods (RMLP with TF and GF) for this highly complex process system.

The performance of the polynomial NARX technique was satisfactory for SSP but a total failure for MSP. The failure of the polynomial NARX in MSP is probably due to the highly complex nature of the UTSG, in addition to the high level of process and sensor noise present in the system.

CNNs were then used for the purpose of designing an empirical UTSG model. The TF method was investigated first. In this technique the individual weight update mode was used since it gave better results than the batch weight update mode in the artificial case-study. Again, similarly to the case-study two data sets were used: the training data set and the testing data set. The CNN was trained on the training data using TF, but the weights were chosen based on their performance on the testing data set using GF. Training was stopped when the MSE on the testing data set started to increase, thus preventing overfitting. The medium power level range for water level was modeled first. A layer and node search were performed using the guidelines outlined in chapter III. The CNN which performed best was the 4-3-2-1 network. Its performance was much better than the performance of the polynomial NARX technique in both SSP and MSP. However, it partially failed to accurately predict the water level response for many other step and ramp power level increases in the training data and the validation data. The partial failure of the CNN in the MSP using the TF method was further investigated by generating noiseless data set using the simulator. A layer and node search were performed again. By removing the noise, the CNN was capable of performing the MSP task. This was an indication that noise had a major impact on CNN design. To prove this point further, an attempt was made to model the high power level range which has a relatively higher signal to noise ratio than the medium power level range. The predictor obtained using CNN TF gave satisfactory results, however, the GF technique gave much better results.

The GF method was then further investigated. The individual weight update mode was used. Similarly to the case-study and the TF method two data sets were used:

the training data and the testing data. The CNN was trained on the training data using GF, and the weights were chosen based on their performance on the testing data using GF, as well. Training was stopped when the MSE on the testing data started to increase, thus preventing overfitting. Using the procedures outlined in chapter III it was determined that one delayed output and two delayed inputs were needed in the network input layer. Note that only two out of the four inputs were delayed for feedback purposes, namely, the feedwater flow rate and the steam flow rate. Prior to performing a node search, a seed search was performed using a small learning rate. The seed search was a critical step since the network was not able to learn any of the training data without the appropriate weight and bias initialization. The search for the number of hidden nodes converged to four nodes in a single hidden layer, for all the six CNN models designed for the UTSG.

The CNN models developed using the GF method gave good performance for MSP and SSP at all power level ranges. To confirm this finding extensive new simulation tests were performed for validating the UTSG CNN models at all power levels. Modeling the low power range was more difficult than modeling the medium and high power ranges. This is due to the reverse thermal dynamic effects, which are more pronounced at the low power levels, and it is also due to the high process and sensor noise which is very significant at low power levels. As a result, the performance of the CNN models at low power levels is not as good as at medium and high power levels. For all the simulations performed, the response for the secondary steam pressure was always better than the response for the downcomer water level. This is due to the fact that the secondary steam pressure is not subject to the reverse dynamic

effects. Overall, the designed UTSG CNN models performed very well throughout the operating range of the UTSG examined, and for various process and sensor noise level.

VI.2 Conclusions and Recommendations

The objective of this research study was to provide a novel method for designing CNN models capable of performing MSP and SSP for complex process system. This objective was achieved by using the RMLP network with GF. A new learning algorithm was derived that takes into consideration the GF in an RMLP network. Network design was achieved by using the guidelines provided for network complexity, and for preventing overfitting.

The conclusions drawn from this research can be summarized as follows:

- (1) GF used along with the RMLP network is an effective structure for modeling complex process systems capable of performing accurate MSP and SSP.
- (2) The power of the derived learning algorithm for training an RMLP with GF lays on the inclusion of the gradients of the CNN predictions at time-step k with respect to the CNN predictions at time step $k - 1$. Without these gradients the RMLP will not learn the dynamics of the system, as it was the case with the UTSG. For less complex system with low noise level, the contribution of these additional gradient terms appear insignificant.
- (3) Excessive noise associated with the operation of the UTSG at low power levels did not prevent the CNNs from learning the dynamics of the system, and as expected, the lower the noise level was the better was the prediction accuracy of the models.

- (4) There is an appropriate number of degrees of freedom needed in the CNN models. in order to accomplish proper generalization. In the UTSG case, one hidden layer with four nodes were sufficient for all the designed predictors. Adding more nodes to the network prevents it from generalizing. The rule of thumb is to start with one or two nodes and increase the number of nodes as needed.
- (5) The notion that the internal recurrency feedback provided by the nodes of the RMLP is by itself sufficient for the network to learn the dynamics of the system under study is not necessarily true. The user must resort to utilizing back past delayed inputs and/or outputs to enhance the learning and increase the prediction accuracy of the CNN in performing MSP. In other words, open-loop identification is not as efficient as closed-loop identification.

For possible further research the accuracy the RMLP with GF algorithm should be increased. This can be achieved by going further back in time in the gradients derivation. This might enable the CNN to approximate more complex systems where noise levels are very high. Another area of investigation would be to apply the RMLP with GF algorithm to the areas of forecasting and prediction. control engineering, as well as condition monitoring and fault diagnosis. Finally, some work must be done in investigating the biologically inspired improvements to the simple artificial neurons utilized throughout this study. This would lead to the design of more effective and intelligent neural networks.

REFERENCES

- [1] A.F. Atiya, Personal Communication, Cairo University, Egypt. 1993.
- [2] R. Beale and T. Jackson. *Neural Computing: An Introduction*. Adam Hilger, Bristol, New York, 1989.
- [3] S.A. Billings, "Identification of Nonlinear Systems - A Survey," *IEEE Proc. Control Theory and Applications*, vol. 5, pp. 272-285, 1980.
- [4] S.A. Billings, S. Chen, and Korenberg, "Identification of MIMO Nonlinear Systems Using a Forward-regression Orthogonal Estimator." *Int. J. Control*, vol. 49, pp. 2157-2189, 1989.
- [5] S.A. Billings and S. Chen. "Extended Model Set. Global Data and Threshold Model Identification of Severely Nonlinear Systems," *Int. J. Control*, vol. 50, pp. 1897-1923, 1989.
- [6] S.A. Billings and Q.H. Tao, "Model Validity Tests for Nonlinear Signal Processing Applications," *Int. J. Control*, vol. 54, pp. 157-194, 1991.
- [7] S.A. Billings, H.B. Jamaluddin. and S. Chen. "Properties of Neural Networks with applications to modelling Non-linear Dynamical Systems," *Int. J. Control*, vol. 55, pp. 193-224, 1992.
- [8] S.A. Billings and W.S.F. Voon, "Structure Detection and Model Validity Tests in the Identification of Nonlinear Systems," *IEEE Proc. D. Control Theory and Applications*, vol. 130, pp. 193-199, 1983.
- [9] S.A. Billings and W.S.F. Voon. "Correlation based Model Validity Tests for Nonlinear Models." *Int. J. Control*, vol. 44, pp. 235-244, 1986.
- [10] S. Chen and S.S. Billings. "Modeling and Analysis of Nonlinear Time Series," *Int. J. Control*, vol. 50, pp. 2151-2171, 1989.
- [11] S. Chen and S.A. Billings, "Representations of Nonlinear Systems: the NAR-MAX Model." *Int. J. Control*, vol. 49, pp. 1013-1032, 1989.
- [12] S. Chen and S.S. Billings. "Recursive Prediction Error Parameter Estimator for Nonlinear Models." *Int. J. Control*, vol. 49, pp. 569-594, 1989.
- [13] S. Chen and S.S. Billings. "Orthogonal Least Squares Methods and their Application to Nonlinear System Identification." *Int. J. Control*, vol. 50, pp. 1873-1896, 1989.

- [14] S. Chen, S.S. Billings, and P.M. Grant, "Nonlinear System Identification Using Neural Networks," *Int. J. Control*, vol. 51, pp. 1191-1214, 1990.
- [15] J.I. Choi, "Nonlinear Digital Computer Control for the Steam Generator System in a Pressurized Water Reactor Plant," Ph.D. dissertation, Dep. Nuclear Eng., Mass. Inst. Technol., Cambridge, MA, 1987.
- [16] K.T. Chong, "Nonlinear Dynamic System Identification Using Neural Networks," Ph.D. dissertation, Texas A&M University, College Station, TX, 1992.
- [17] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function", *Mathematics of Control, Signals, and Systems* vol. II, pp. 133-141, June 1990.
- [18] P. Eykhoff, *System Identification*. John Wiley & Sons, Inc. New York, NY, 1974.
- [19] S.Y. Fakhouri, "Identification of the Volterra Kernels of Nonlinear Systems," in *IEEE Proc. D, Control Theory and Applications*, vol. 127, pp. 296-304, 1980.
- [20] B. Fernandez, A.G. Parlos, and W.K. Tsai, "Nonlinear System Identification using Artificial Neural Networks," *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. II, pp. 133-141, June 1990.
- [21] W.F. Ganong, *Review of Medical Physiology*. Lange Medical Publication, Los Altos, CA, 1985.
- [22] C.E. Garcia, D.M. Prett, and M. Morari, "Model Predictive Control: Theory and Practice- A Survey," *Automatica*, vol. 25, pp. 335-348, 1989.
- [23] R. Harber and H. Unbehauen. "Structure Identification of Nonlinear Dynamic Systems - A Survey on Input/Output Approaches," *Automatica*, vol. 26, No 4-A, pp. 651-677, 1990.
- [24] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to The Theory of Neural Computation*. Addison-Wesley Publishing Co., Reading, MA, 1991.
- [25] W.W. Hines and D.C. Montgomery, *Probability and Statistics in Engineering and Management Science*. pp. 213-216, John Wiley & Son, Inc. New York, NY, 1980.
- [26] D.T. Horak, "Experimental Identification of Modeling Errors in Dynamic Systems," *American Control Conference*, vol. 2, pp. 1307-1312, 1988.
- [27] R. Isermann, "Process Fault Detection Based on Modeling and Estimation Methods-A Survey," *Automatica*, vol. 20, pp. 387-404, 1984.

- [28] A. Juditsky, Q. Zhang, B. Delyon, and A. Benveniste, "Wavelets In Identification," Tutorial, May 1994.
- [29] A. Krogh and J. Hertz, *A Simple Weight Decay Can Improve Generalization*. Advances in Neural Information Processing Systems 4, pp. 450-957, Morgan Kaufmann Publishers, San Mateo, CA, 1992.
- [30] S.W. Kuffler, J.G. Nicholls, and A.R. Martin. *From Neuron to Brain*. Sinauer Associates Inc., Sunderland, MA, 1984.
- [31] I.J. Leontaritis and S.A. Billings, "Model Selection and Validation Methods for Nonlinear Systems," *Int. J. Control*, vol. 45, pp. 311-341, 1987.
- [32] L. Ljung and S. Gunnarsson, "Adaptation and Tracking in System Identification - A Survey," *Automatica*, vol. 26, pp. 7-21. 1990.
- [33] L. Ljung and T. Sjöberg, "A System Identification Perspective on Neural Nets," *IEEE Workshop on Neural Networks for Signal Processing*, May 1992.
- [34] L. Ljung, *System Identification Theory for the Users*. Prentice Hall, Hertfordshire, UK, 1987.
- [35] L. Ljung and T. Glad, *Modeling of Dynamic Systems*. Prentice Hall, Hertfordshire, UK, 1994.
- [36] R.J. MacGregor. *Neural and Brain Modeling*. Academic Press, Inc. San Diego, CA, 1987.
- [37] K.S. Narendra and K. Parthasarathy, "Identification and Control of Dynamic System Using Neural Networks." *IEEE Transactions on Neural Networks*, vol. 1. pp. 4-27. 1990.
- [38] K.S. Narendra and K. Parthasarathy, "Neural Networks in Control Systems," Workshop on Neural Networks in Control Systems. *American Control Conference*, Workshop Manual, 1991.
- [39] K.S. Narendra and W.L. Parthasarathy, "Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252-262. 1991.
- [40] D.C. Park, M.A. EL-Sharkawi, R.J. Marks II, L.E. Atlas, and M.J. Damborg "Electric Load Forecasting using An Artificial Neural Networks." *IEEE Transactions on Power Systems*, vol. 6, pp. 442-449. May 1991.

- [41] A.G. Parlos, A. Atiya, K.T. Chong, B. Fernandez, and Wei Tsai, "Nonlinear Identification of Process Dynamics Using Neural Networks," *Nuclear Technology*, vol. 97, pp. 79-95, 1992.
- [42] A.G. Parlos, K.T. Chong, and A. Atiya, "Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics," *IEEE Transactions on Neural Networks*, vol.5, pp. 255-266, 1994.
- [44] A.G. Parlos, K.T. Chong, and A. Atiya, "Dynamic Gradient Descent Learning Algorithms for Enhanced Empirical Modeling of Power Plants," *ANS Winter Annual Meeting*, San Francisco, CA, vol. 64, pp. 178-179, Nov. 1991.
- [45] A.G. Parlos, K.T. Chong, and A. Atiya, "U-Tube Steam Generator Empirical Model Development and Validation Using Neural Networks," *ANS Transactions*, Boston, MA., vol. 65, pp. 108-109, June 1992.
- [46] A.G. Parlos, B. Fernandez, A. Atiya, J. Muthusami, and W.K. Tsai, "An Accelerated Learning Algorithm for Multilayer Perceptron Networks," *IEEE Transactions on Neural Networks*, vol.5, pp. 493-497, 1994.
- [47] S. Parthasarathy, "Model Adaptive Control of Process Systems Using Recurrent Neural Networks," M.S. Thesis, Texas A&M University, College Station, TX, 1993.
- [48] R. Patton, P. Frank, and R. Clark, "Fault Diagnosis in Dynamic Systems," *Theory and Application*. Prentice Hall, Hertfordshire, UK, 1989.
- [49] R.S. Pinsky and D.L. Rubinfeld, *Econometric Models and Economic Forecasting*. McGraw-Hill, Inc., New York, NY, 1991.
- [50] F.J. Pineda, "Generalization of Backpropagation to Recurrent Neural Networks," *Physical Review Letters*, vol. 59, pp. 2229-2232, 1987.
- [51] F.J. Pineda, "Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation," *Neural Computation*, vol. 1, pp. 161-172, 1989.
- [52] J.C. Principe, J.Kuo, and S.Celebi, "An Analysis of the Gamma Memory in Dynamic Neural Networks," Brief Paper, *IEEE Transactions on Neural Networks*, vol 5, March 1994.
- [53] F. Rosenblatt, "The Perceptron; A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Reviews*, vol. 65, pp. 386, 1958.

- [54] T.H. Ruch. H.D. Patton, *Physiology and Biophysics*. W. B. Saunders Co., Philadelphia. PA. 1965.
- [55] J. Sejnowski, C.R. Rosenberg, "Parallel Networks that learn to pronounce English text," *In Complex Systems*, pp. 145-168, 1987.
- [56] Hong-Te Su, T.J. McAVoy, and P. Werbos, "Long Term Predictions of Chemical Processes Using Recurrent Neural Networks : A Parallel Training Approach," *Ind. Eng. Chem. Res.*, vol. 31, pp. 1338-1352, 1992.
- [57] K.P. Simpson, *A Review of Artificial Neural System I: Foundations*. General Dynamics Electronics Division. San Diego, CA, May 1988.
- [58] T. Sjöberg and L. Ljung, "Criterion Minimization Using Estimation Data And Validation Data." Tutorial. 1994.
- [59] T. Söderström and Stoica, *System Identification*. Prentice Hall, Hertfordshire, UK, 1988.
- [60] W.H. Strohmayer, "Dynamic Modeling of Vertical U-Tube Steam Generators for Operational Safety Systems," Ph.D. dissertation, MIT, Cambridge, MA, Aug. 1982.
- [61] V. Strejc, "Least Squares Parameter Estimation," *Automatica*, vol. 16, pp. 535-550, 1980.
- [62] S. Menon. "Gain-scheduled Nonlinear Control of U-Tube Steam Generators at Low Powers." M.S. Thesis. Texas A&M University, College Station. TX, 1991.
- [63] E. Tzirkel-Hancock and F. Fallside, "A Direct Control Method for a Class of Nonlinear Systems Using Neural Networks," Technical report. Cambridge University Engineering Department. Cambridge, UK. 1991.
- [64] P.J. Werbos. "Backpropagation Through Time: What it Does and How to Do It," *Proceedings of the IEEE*, vol. 78, pp. 1550-1560, 1990.
- [65] P.J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. dissertation. Harvard University, 1974.
- [66] D.A White and D.A. Sofge, *Handbook of Intelligent Control*. Van Nostrand Reinhold. New York, 1992.
- [67] N. Wiener. *Cybernetics : or Control and Communication in the Animal and the Machine*. MIT Press, Cambridge, 1948.

- [68] R.J. Williams and D. Zipser, "A Learning Algorithm for Continually Running fully Recurrent Neural Networks," *Neural Computation*, vol. 1 pp. 270-280, 1989.
- [69] R.J. Williams and J. Peng, "An efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories," *Neural Computation*, vol. 2, pp. 490-501, 1990.

Year	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100
1970	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100

THE FUNCTIONAL NEURON

In this appendix the functional neuron used in CNNs throughout this study is discussed. This functional neuron is also known as the perceptron.

The perceptron model was first introduced by Rosenblatt [53], and since its introduction it has been studied extensively. The original perceptron is a heteroassociative, nearest-neighbor pattern matcher that stores the M pattern pairs (\mathbf{X}_k, y_k) , $k = 1, 2, \dots, M$, using some form of a supervised learning algorithm. The elements of the vector \mathbf{X}_k take analog values, whereas the scalar y_k takes values of either -1 or $+1$. The equations governing the operation of the perceptron can be expressed as follows:

$$y_k = F\left(\sum_{i=1}^N w_i x_k^i + w_0\right) = F(\mathbf{W}^T \cdot \mathbf{X}_k), \quad (109)$$

where,

$$\mathbf{W} = \begin{pmatrix} w_1 \\ \vdots \\ w_N \\ w_0 \end{pmatrix}, \quad \mathbf{X}_k = \begin{pmatrix} x_k^1 \\ \vdots \\ x_k^N \\ 1 \end{pmatrix}, \quad (110)$$

where $F(\cdot)$ is the threshold function, and where n is the size of the input vector for each pattern k .

The perceptron had some early success in the field of CNN. However, it suffered from some deficiencies which were published during the sixties and early seventies. This was to change in the 1980s. The discontinuous nonlinear function (i.e, the threshold function) $F(\cdot)$ that featured in the earlier perceptron model was replaced

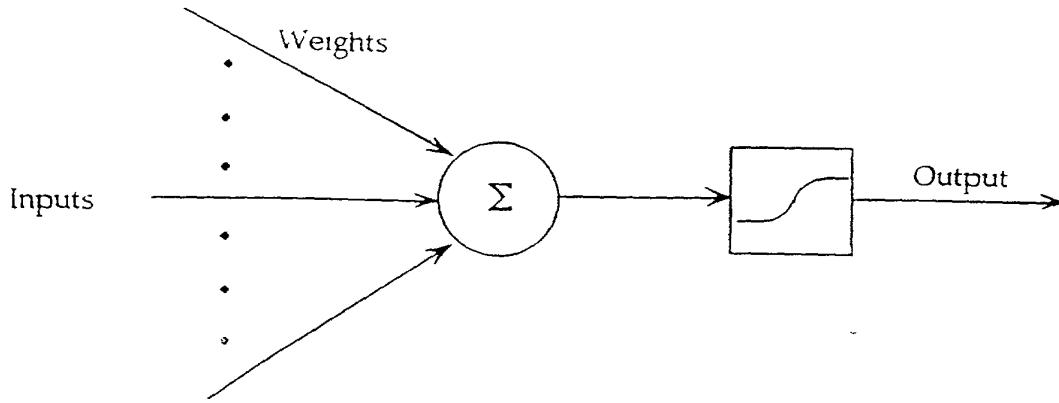


Figure 80. A Functional Neuron (Perceptron).

by a smooth function which approximated it. The perceptron architecture is shown in Figure 80. The nonlinear function $F(\cdot)$ is termed variously as activation function, transfer function, or squashing function. A typical example of such a squashing function is :

$$F(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}. \quad (111)$$

The limits of this function are -1 and 1 for $x \rightarrow -\infty$ and $x \rightarrow \infty$, respectively. This limits are the same as that of the original threshold function, however, because of the smooth nature of $F(x)$, the gradient of the output y_k with respect to the parameters of the CNN can now be computed. A large fraction of current research and development in CNNs is based on this functional neuron.

310
ABSTRACT

Adaptive Filtering in Complex Process Systems

Using Recurrent Neural Networks. (August 1996)

Sunil Kumar Menon, B. S., Regional Engineering

College, Trichy-India; M. S., Texas A&M University

Chair of Advisory Committee: Dr. Alexander G. Parlos

The objective of this research study is to develop a method for adaptive state filtering in complex process systems using recurrent neural networks (NN). Nonlinear and “nonparametric” adaptive state filtering methods, such as the one developed here, have many applications in industrial operations and maintenance.

The proposed nonlinear and “nonparametric” state filtering method is based on the fundamental principles of the Kalman Filter. However, in contrast to conventional state filtering methods which make use of linear process models, minimal assumptions are placed upon the process model used in the developed filtering method. The key process model assumptions stem from the limitations of the NNs to approximate arbitrary nonlinear functions. Further, no assumptions are placed on the process model noise. In fact, the developed method formulates a minimum variance filter, which is designed using least-squares algorithms. Specifically, nonadaptive and adaptive forms of the proposed state filtering method is developed, and its applicability investigated. Additionally, a hybrid form of the same method is developed, which uses both the nonadaptive and adaptive developments.

The proposed method is applied to three simulated process systems, with increasing levels of complexity: an artificial problem, a DC Motor-Pump system, and a U-Tube Steam Generator (UTSG) system. In addition to estimating dynamic states, the proposed filtering method is used to estimate critical parameters of a DC Motor-Pump and a UTSG system. It is found that the proposed filtering method performs

well in all cases studied. As anticipated, the accuracy of the state estimates are directly dependent upon the fidelity of the process models used in the filter development.

This research study demonstrates that certain NNs can be effectively used for adaptive nonlinear state filtering, when very little is known explicitly about the dynamics of the process under consideration. This conclusion complements the well-known capabilities of certain NNs in modeling the input-output behavior of complex systems, and it is, again, attributed to their superior function approximation capability. Experimental verification of these simulation results is warranted, and it should be pursued as the next step towards building confidence in these nonlinear computational tools.

ACCEPTED MANUSCRIPT

ACKNOWLEDGEMENTS

I would like to thank Dr. Alexander G. Parlos, my research advisor, for his tireless support and encouragement throughout the course of this research. I was very fortunate to have worked under the supervision of Dr. Parlos and witness his dedication to research as well as to his students. I would also like to thank my other committee members, Dr. Paul Nelson, Dr. Fred Best, and Dr. Karen Butler for their support and advice.

I would also like to thank my colleagues, Igor Carron, Mark Shavers, Esmacil Oufi, Omar Rais, and Rube Williams for their assistance and encouragement.

TABLE OF CONTENTS

CHAPTER	Page
I INTRODUCTION	1
I.1 A Brief Note on Complex Systems	2
I.2 Fundamental Problems in Dynamic Systems	3
I.3 An Overview of Estimation Problems	5
I.4 Industrial Estimation Problems	13
I.5 The Role of Neural Networks in Estimation	17
I.6 Literature Review	20
I.7 Research Contributions	27
I.8 Organization of Dissertation	28
II AN OVERVIEW OF LINEAR ESTIMATION METHODS	30
II.1 Introduction	30
II.2 System Models	31
II.3 Nonadaptive Methods	34
II.4 System Identification	44
II.5 Adaptive Methods	49
II.6 Chapter Summary	49
III NONLINEAR ESTIMATION METHODS-PREDICTION	51
III.1 Introduction	51
III.2 System Models	52
III.3 Nonlinear Prediction: A General Framework	55
III.4 Conventional Expansions	56
III.5 Neural Network Expansions	62
III.6 Chapter Summary	94
IV NONLINEAR ESTIMATION METHODS - STATE FILTERING	95
IV.1 Introduction	95
IV.2 Nonlinear State Filtering: A General Framework	96
IV.3 Conventional Expansions	98
IV.4 Neural Network Expansions	104
IV.5 Neural Network State Filter Performance Evaluation	119
IV.6 Chapter Summary	121
V APPLICATION 1 - TWO-INPUT-TWO-OUTPUT SYSTEM	122

V.1	Introduction	122
V.2	System Description	122
V.3	System Model Description	123
V.4	Neural Network State Filters	125
V.5	Validation Results	134
V.6	Chapter Summary	148
VI	APPLICATION 2 - DC MOTOR-PUMP SYSTEM	153
VI.1	Introduction	153
VI.2	Process Description	153
VI.3	Process Model Description	158
VI.4	Neural Network State Filters	160
VI.5	Validation Results	176
VI.6	Chapter Summary	196
VII	APPLICATION 3 - U-TUBE STEAM GENERATOR SYSTEM	200
VII.1	Introduction	200
VII.2	Process Description	201
VII.3	Process Model Description	205
VII.4	Neural Network State Filters	207
VII.5	Validation Results	231
VII.6	Chapter Summary	278
VIII	SUMMARY AND CONCLUSIONS	294
VIII.1	Summary of the Research Study	294
VIII.2	Conclusions from the Research Study	302
VIII.3	Recommendations for Future Research Directions	303
REFERENCES	305
APPENDIX		
A	BACKPROPAGATION-THROUGH-TIME ALGORITHM	313
A.1	Introduction	313
A.2	The BTT Algorithm	314
VITA	318

LIST OF TABLES

TABLE	Page
1 2I2O System Estimation Data Set Used by the Nonadaptive and Adaptive NN State Filters.	126
2 2I2O System Validation Data Set Used by the Nonadaptive and Adaptive NN State Filters.	137
3 2I2O System Nonadaptive and Adaptive NN State Filter Results.	152
4 Motor-Pump System Estimation and Validation Data Set Used by the Hybrid Adaptive and Adaptive NN Armature Resistance Filters.	161
5 Motor-Pump System Estimation and Validation Data Set Used by the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter.	162
6 Motor-Pump System Hybrid Adaptive and Adaptive NN Armature Resistance Filter Validation Test Results.	199
7 Motor-Pump System Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Validation Test Results.	199
8 UTSG Process System Estimation and Validation Data Set Used by the Hybrid Adaptive and Adaptive NN Riser Void Fraction Filters 1,2 and 3.	208
9 UTSG Process System Estimation and Validation Data Sets Used by the Adaptive NN HTC Filter 4.	209
10 UTSG Process System Hybrid Adaptive NN Riser Void Fraction Filter 1 Validation Test Results.	292
11 UTSG Process System Hybrid Adaptive NN Riser Void Fraction Filter 2 Validation Test Results.	292
12 UTSG Process System Adaptive NN Riser Void Fraction Filter 3 Validation Test Results.	293
13 UTSG Process System Adaptive NN HTC Filter 4 Validation Test Results.	293

LIST OF FIGURES

FIGURE		Page
1	Schematic Representation of a Complex System.	3
2	Types of State Estimation Problems.	7
3	Multi-Step-Ahead Prediction Schematic on a p-Step Horizon.	9
4	Block Diagram of a SSP.	10
5	Block Diagram of a State Filter.	11
6	Block Diagram of an Inferential Control Scheme.	15
7	Block Diagram of an Adaptive Control Scheme.	16
8	Block Diagram of a Fault Diagnosis Scheme Using Estimation.	17
9	Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARX Predictors.	39
10	Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARMAX Predictors.	41
11	Kalman Filter Timing Diagram.	44
12	Schematic Diagram of the Kalman Filter.	45
13	Block Diagram of Adaptive Filtering Using System Identification.	46
14	Block Diagram Representation of the Single-Step-Ahead and Multi-Step-Ahead NARX Predictors.	57
15	Schematic Diagram of the FMLP.	63
16	Schematic Diagram of the RMLP.	65
17	A Neural Network Predictor Representation.	68
18	Graphical Representation of the FMLP Sublayers.	71

FIGURE		Page
19	Block Diagrams of the FMLP Single-Step-Ahead and Multi-Step-Ahead Predictors.	73
20	Graphical Representation of the RMLP Sublayers.	76
21	Block Diagrams of the RMLP Single-Step-Ahead and Multi-Step-Ahead Predictors.	79
22	Block Diagrams of NN Single-Step-Ahead and Multi-Step-Ahead Predictors.	81
23	Diagram of the Estimation and Validation Data Sets Used to Develop a NN.	91
24	Block Diagram of a Nonlinear State Filter.	99
25	Timing Diagram of the Extended Kalman Filter.	102
26	Block Diagram of a Nonadaptive Neural Network State Filter.	108
27	Block Diagram Showing the Relation between the Predictors Used in the Nonadaptive and Adaptive Neural Network State Filters.	115
28	Block Diagram of an Adaptive Neural Network State Filter.	117
29	Block Diagram of a Hybrid Adaptive Neural Network State Filter.	120
30	Block Diagram of the 2I2O System Nonadaptive NN State Filter.	128
31	2I2O System Output Response, from the Simulator and System Model 1 Predictor, for the Nonadaptive NN State Filter Using the Estimation Data Set as Inputs.	129
32	2I2O System State $x_3(t)$ Response, from the Simulator, Model 1, and NN State Filter, for the Nonadaptive NN State Filter Using the Estimation Data Set as Inputs.	130
33	2I2O System Output Response, from the Simulator and the NN Output, for the Adaptive NN State Filter Using the Estimation Data Set as Inputs.	133
34	Block Diagram of the 2I2O System Adaptive NN State Filter.	135

35	2I2O System State $x_3(t)$ Response, from the Simulator, Model 2, and NN State Filter, for the Adaptive NN State Filter Using the Estimation Data Set as Inputs.	136
36	2I2O System Output Response, from the Simulator and the Model Predictor, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).	138
37	2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).	139
38	2I2O State Filter Normalized Residuals for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).	140
39	2I2O System Output Response, from the Simulator and the Model Predictor, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).	142
40	2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).	143
41	2I2O State Filter Normalized Residuals Values for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).	144
42	2I2O System Output Response, from the Simulator and the NN Output Predictor, for the Adaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).	145
43	2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter for the Adaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).	146
44	2I2O State Filter Normalized Residuals for the Adaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).	147
45	2I2O System Output Response, from the Simulator and the NN Output Predictor, for the Adaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).	149

46	2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter, for the Adaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).	150
47	2I2O State Filter Normalized Residuals for the Adaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment). . . .	151
48	Schematic Diagram of the DC Motor, Centrifugal Pump and Associated Piping System.	154
49	Block Diagram of the Motor-Pump Process System.	158
50	Block Diagram of the Motor-Pump System Hybrid Adaptive NN Armature Resistance Filter.	165
51	Motor-Pump System Output Response, from the Simulator and System Model, for the Hybrid Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.	166
52	Motor-Pump System Armature Resistance Response, from the Simulator, Process Model, and the NN State Filter, for the Hybrid Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.	167
53	Block Diagram of the Motor-Pump System NN Output Predictor Used in the Adaptive NN Armature Resistance Filter.	170
54	Motor-Pump System Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.	171
55	Block Diagram of the Motor-Pump System Adaptive NN Armature Resistance Filter.	173
56	Motor-Pump System Armature Resistance Response, from the Simulator, Process Model, and NN State Filter, for the Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs. . . .	174
57	Block Diagram of the Motor-Pump System Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter.	177

58	Motor-Pump System Output Response, from the Simulator and Model Predictor, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Estimation Data Set as Inputs.	178
59	Motor-Pump System Armature Resistance and Flux Linkage Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Estimation Data Set as Inputs.	179
60	Motor-Pump System Output Response, from the Simulator and the Model Predictor, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).	181
61	Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).	182
62	Motor-Pump System Armature Resistance Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).	183
63	Motor-Pump System Output Response, from the Simulator and the Model Predictor, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).	184
64	Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).	185
65	Motor-Pump System Armature Resistance Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).	186
66	Motor-Pump System Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment). . . .	187

67	Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment). . . .	188
68	Motor-Pump System Armature Resistance Filter Normalized Residuals for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).	190
69	Motor-Pump System Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment). . . .	191
70	Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment). . . .	192
71	Motor-Pump System Armature Resistance Filter Normalized Residuals for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).	193
72	Motor-Pump System Armature Resistance and Flux Linkage Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (Low Noise Environment).	194
73	Motor-Pump System Armature Resistance and Flux Linkage Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (Low Noise Environment).	195
74	Motor-Pump System Armature Resistance and Flux Linkage Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (High Noise Environment).	197
75	Motor-Pump System Armature Resistance and Flux Linkage Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (High Noise Environment).	198
76	Schematic Diagram of the U-Tube Steam Generator.	202

77	Block Diagram of the UTSG Process Simulator.	203
78	Block Diagram of the UTSG Process Hybrid Adaptive NN Riser Void Fraction Filter 1.	213
79	UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using the Estimation Data Set as Inputs.	214
80	UTSG Process Riser Void Fraction Response, from the Simulator, Scheduled Model, and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using the Estimation Data Set as Inputs.	215
81	Block Diagram of the UTSG Process Hybrid Adaptive NN Riser Void Fraction Filter 2.	218
82	UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using the Estimation Data Set as Inputs.	219
83	UTSG Process Riser Void Fraction Response, from the Simulator, Simulator Model, and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using the Estimation Data Set as Inputs.	220
84	UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using the Estimation Data Set as Inputs.	222
85	Block Diagram of the UTSG Process Adaptive NN Riser Void Fraction Filter 3.	225
86	UTSG Process Riser Void Fraction Response, from the Simulator, Simulator Model, and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using the Estimation Data Set as Inputs.	226
87	Block Diagram of the UTSG Process NN Output Predictor Used in the Adaptive NN HTC Filter 4.	229
88	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using the Estimation Data Set as Inputs.	230

89	Block Diagram of the UTSG Process Adaptive NN HTC Filter 4.	232
90	UTSG Process HTC, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using the Estimation Data Set as Inputs. . .	233
91	UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (Low Noise Environment).	235
92	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (Low Noise Environment).	236
93	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 with a Ramp Input (Low Noise Environment).	237
94	UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (Low Noise Environment).	238
95	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (Low Noise Environment).	239
96	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (Low Noise Environment).	240
97	UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (High Noise Environment).	241
98	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (High Noise Environment).	242
99	UTSG Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (High Noise Environment).	244

100	UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (High Noise Environment).	245
101	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (High Noise Environment).	246
102	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (High Noise Environment).	247
103	UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (Low Noise Environment).	248
104	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (Low Noise Environment).	249
105	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (Low Noise Environment).	250
106	UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (Low Noise Environment).	251
107	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (Low Noise Environment).	252
108	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (Low Noise Environment).	253
109	UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (High Noise Environment).	254

110	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (High Noise Environment).	255
111	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (High Noise Environment).	257
112	UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (High Noise Environment).	258
113	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (High Noise Environment).	259
114	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (High Noise Environment).	260
115	UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (Low Noise Environment).	261
116	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (Low Noise Environment).	262
117	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (Low Noise Environment).	263
118	UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (Low Noise Environment).	264
119	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (Low Noise Environment).	265

120	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (Low Noise Environment).	266
121	UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (High Noise Environment).	268
122	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (High Noise Environment).	269
123	UTSG Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (High Noise Environment).	270
124	UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (High Noise Environment).	271
125	UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (High Noise Environment).	272
126	UTSG Process Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (High Noise Environment).	273
127	UTSG Process HTC, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 1 (Steady-State at 5% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	274
128	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 2 (Steady-State at 10% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	275
129	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 3 (Steady-State at 15% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	276

130	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 4 (Steady-State at 20% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	277
131	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and Low Noise Environment).	279
132	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	280
133	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 6 (Step Input and Low Noise Environment).	281
134	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 6 (Step Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	282
135	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 7 (Ramp Input and Low Noise Environment).	283
136	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 7 (Ramp Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	284
137	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 8 (Step Input and Low Noise Environment).	285
138	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 8 (Step Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	286
139	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and High Noise Environment).	287

140	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and High Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	288
141	UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 6 (Step Input and High Noise Environment).	289
142	UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 6 (Step Input and High Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)	290
143	Simple Feedforward Neural Network with Global Feedback.	314
144	Unfolding of the Neural Network Across Time.	317

INTRODUCTION

The Industrial Revolution of the 1800s, saw the beginning of a new era in the increased sophistication of industrial processes intended to produce inexpensive products at a massive scale. Similarly, in the 1990s, we are witnessing a new era in the design and operation of “smart” industrial products and processes which is bound to change every facets of our everyday life. In other words, we are continuously witnessing an ever increasing application of complex information processing and decision making in the operation of industrial and consumer products and processes. As a result of this trend, principles from systems theory and operations research are becoming an integral part of the planning, design, operation, and maintenance of modern products and processes. For the remainder of this dissertation, technical products and processes will be referred to simply as “systems”.

For the purposes of this study, the systems examined will be assumed to belong to one of the following two categories:

- (1) Mechanical, electrical, and electro-mechanical systems, e.g. electric machines, cars, etc., including systems whose principal behavior is governed by rigid-body dynamics; i.e. systems necessitating use of the laws of conservation of momentum and angular momentum for adequate description,
- (2) Process systems, e.g. chemical and nuclear reactors, manufacturing processes, etc.; i.e. systems necessitating use of the laws of thermodynamics, fluid flow and heat transfer for adequate description.

The majority of the developments presented in this dissertation are applicable to both electro-mechanical (electrical and mechanical aspects) and process systems, while the latter being our focus of interest. In particular, a process system is defined as an assembly of physical components which function together to produce a

This dissertation follows a style based on the *IEEE Transactions on Automatic Control*.

specific value-added product. For example, the product might be a commodity such as electricity or a certain chemical produced by a chemical process, or even an integrated circuit chip resulting from several stages in microelectronics manufacturing. The single most important characteristic of the systems considered in this study is their “complexity”, and its impact on our ability to operate them effectively.

I.1 A Brief Note on Complex Systems

A complex system, as used in this study, is characterized by the following features [22]:

- (1) Dynamic behavior, i.e. its measured response exhibits significant memory,
- (2) Nonlinear behavior, i.e. its measured response exhibits strong nonlinearities and the linear superposition principle is no longer applicable,
- (3) Multivariable behavior, i.e. multiple inputs, multiple disturbances, multiple states, and multiple outputs are needed to sufficiently describe its behavior,
- (4) High, frequently infinite, dimensionality, i.e. it is governed by laws of physics and/or chemistry which are expressed as partial differential equations,
- (5) Stochastic behavior, i.e. the stochastic component of its response is a significant part of its overall measured response, and,
- (6) Poorly understood and/or not well-understood behavior, i.e. the physics and/or chemistry governing its behavior are not well understood, and therefore no validated “first-principles” model is available.

The aforementioned features imply that as the “complexity” of a system increases, our ability to “model” it diminishes. Therefore, our focus in using neural networks (NNs) for estimation, and for filtering, in particular, is to concentrate on systems that are difficult to “model”, though easy to “simulate”. Representative examples of such systems are complex electromechanical and process systems, widely encountered in manufacturing, for example, operations.

Certain electromechanical and even process systems may not be considered “complex” during normal operations. However, during failure modes and as a result

of wear and tear effects, such systems must be considered complex because they satisfy all of the aforementioned characteristics. As a result of this characterization of complex systems, we do not assume nor do we make use of any specific representation for their description. Rather, it is assumed that either samples from a real hardware set-up are available or that a validated simulator takes on the role of such a set-up. A schematic representation of a complex system, as utilized in this study, is depicted in Figure 1.

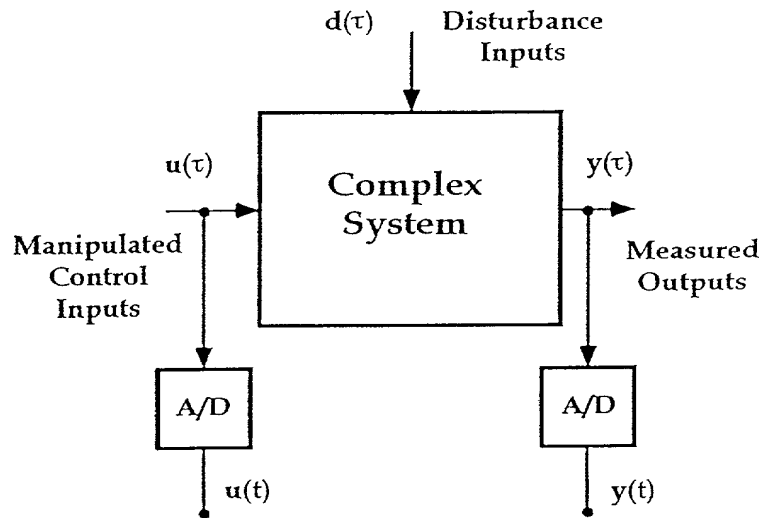


Figure 1. Schematic Representation of a Complex System.

I.2 Fundamental Problems in Dynamic Systems

In dealing with dynamic systems, four classes of fundamental problems can be dealt with. These are as follows [69]: *modeling*, *analysis*, *estimation*, and *control*. These problems are briefly described in the following paragraphs:

Modeling refers to the development of mathematical models of systems and it is the most important step in the solution of a vast number of engineering problems [44]. In general, there are two types of mathematical models of dynamical systems: *physical models*, also known as white-box models, which are developed using the laws of physics (first-principles), such as conservation of mass, momentum, and energy and/or laws of chemistry; and *empirical models*, also known as black-box models, which are developed using measurements (observations) collected from the system under study. In practice, however, physical models of complex systems incorporate some amount of empiricism (look-up tables, algebraic fits etc.), because of the difficulty in explaining some physical phenomena using only first-principles. Therefore, a third category of models, known as grey-box models, arises. The complexity of mathematical models can range from a highly sophisticated physical model requiring extensive computations, to an empirical model with a few algebraic equations. The type of a mathematical model and the required level of complexity for a specific application is dictated by the needs of the application itself. For example, extremely sophisticated physical models may be required for design applications, while simpler and less accurate empirical models may suffice for control applications. The latter however, do not provide the physical insight needed in system design.

Analysis deals with the subject of qualitatively and/or quantitatively describing the behavior of the system outputs. A qualitative analysis of the system describes its general properties, such as stability, deviation from set-points, etc. . A quantitative analysis describes more detailed aspects of the systems, such as temporal behavior, dominant frequencies of the system output, etc. . Again, as in modeling, a qualitative and/or quantitative analysis is done depending on the specific objectives for which the analysis is undertaken. However, as the complexity of the system being modeled increases, computational system analysis becomes one of the few available choices.

Estimation is the process of inferring the value(s) of a variable of interest in a system, using the measurements (observations) collected from the same system. In other words *estimation* is the process of extracting information from data. This definition of estimation is quite similar to the definition of *inductive learning* used in

various fields of computer science. *Estimation* of a quantity or a variable can take numerous forms depending on the problems being studied. In particular, when dealing with dynamic systems, estimation problems can be classified into three types: *state estimation*, *system identification*, and *adaptive estimation*. More detailed treatment of each of these problems is given in the following subsection.

Control refers to the process of determining the values of the manipulated system inputs (control inputs) that cause its outputs to behave, as close, in a pre-determined manner, as possible. The purpose of control systems could be to stabilize the system (cause the system outputs to remain bounded within certain limits when bounded input signals are applied) and/or achieve performance, such as tracking (cause the system outputs to follow a predetermined trajectory). Furthermore, two types of control problems can be defined: *open-loop control* and *closed-loop control*. In open-loop control the control inputs are pre-determined using the system's initial conditions only. In closed-loop control the control inputs are computed based on the system output measurements. This is also referred to as *feedback control*. Even though open-loop control systems are widely used in industry, especially in manufacturing processes, whenever possible feedback control is preferable for two reasons: (1) to stabilize possibly unstable open-loop systems, and, (2) to reduce the effects uncertainties have on the overall system performance.

I.3 An Overview of Estimation Problems

Estimation problems in dynamic systems can be generally classified into three categories, as follows [69]:

- (1) *State Estimation* is the process of inferring the *state(s)* of the system using the measurements (observations) collected from the system itself, and a pre-specified mathematical model of the system. The states of the system are variables that completely specify its behavior (assuming that the system model used is an accurate enough reflection of the physical phenomena governing its behavior).

- (2) *System Identification* (SI) is the process of determining a mathematical model of a system using the measurements collected from the system itself. These type of mathematical models, as explained earlier, are called *empirical* or *black-box* models.
- (3) *Adaptive State Estimation* is the process of state estimation when a system model is not available. Therefore, the system model must be identified first, using SI techniques, and then state estimation can be performed. Thus, adaptive estimation is actually a combination of the problems of state estimation and ~~system identification~~.

I.3.1 State Estimation

In further attempting to define and refine state estimation, several categories of this problem can be defined [28]. The notation used, here and throughout the manuscript, for the state estimate, $\hat{\mathbf{x}}(t|t)$, denotes its value at a discrete time t based on measured sensor information up to and including the discrete time instant t . All developments in this study are presented in the discrete-time domain. Depending upon the time instant for which a value for the state estimate, $\hat{\mathbf{x}}(t|t)$, of the state, $\mathbf{x}(t)$, is desired and the time instant until which measurements, $\mathbf{y}(t)$, are available and/or used, the following three estimation problems can be defined:

- (1) *Smoothing* : given the measurements, $\mathbf{y}(t + \lambda)$, up to and including the time instant $(t + \lambda)$, $\lambda > 0$, the state estimate $\hat{\mathbf{x}}(t|t + \lambda)$ at a past time t is determined.
- (2) *Filtering* : given the measurements, $\mathbf{y}(t)$, up to and including the time instant t , the state estimate $\hat{\mathbf{x}}(t|t)$, at time t is determined.
- (3) *Prediction* : given the measurements $\mathbf{y}(t - \lambda)$, up to and including the time instant $(t - \lambda)$, $\lambda > 0$, the state estimate $\hat{\mathbf{x}}(t|t - \lambda)$, at the future time t is determined. If $\lambda = 1$, this is referred to as *single-step-ahead prediction*, and if $\lambda = p$, where $p > 1$, this is referred to as *p-step-ahead* or *multi-step-ahead prediction*.

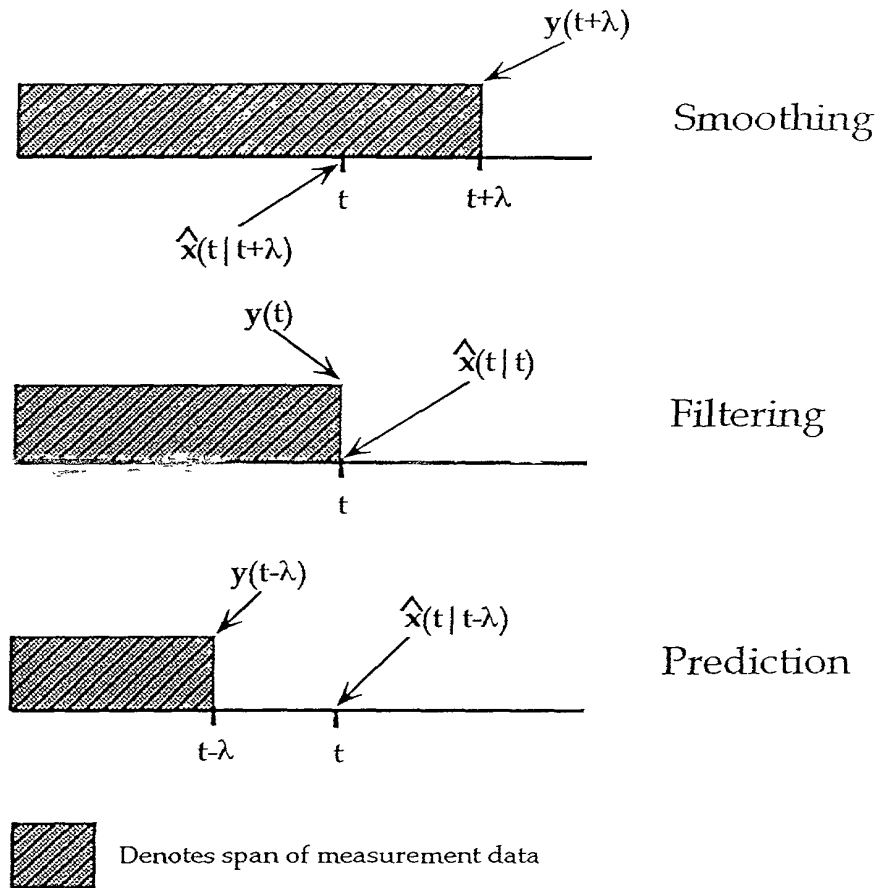


Figure 2. Types of State Estimation Problems.

Here, it is assumed that there exists a casual relation between the measurements, $y(t)$, and the actual state, $x(t)$, to be estimated. The nature of this relationship dictates the complexity of the estimation problem to be solved, and the strength of the relationship introduces the notions of observability and state reconstructability, issues not dealt with in this research study. The aforementioned three types of state estimation problems are depicted in Figure 2.

In this study, only *filtering* and *prediction* problems are considered, because these are of greater interest for the applications under consideration than *smoothing*

problems. Furthermore, we first consider prediction, followed by filtering, as the former helps to implement the latter.

Prediction

Prediction refers to the estimation of a variable of interest (such as an output, a state etc.) at a future time, based on the measurements available up to and including the current time. Most often the variable of interest is a measured output, $y(t)$, though predicting future values of states, $x(t)$, not being measured is feasible but very challenging.

In general, two types of prediction problems can be formulated: *Single-Step-Ahead Prediction* (SSP- also used for single-step predictor) and *Multi-Step-Ahead Prediction* (MSP-also used for multi-step predictor). SSP is the estimation of a variable at some time-step $(t + 1)$, based on system measurements up to and including the time step t . MSP is the estimation of a variable at some time-step $(t + 1)$, based on system measurements up to and including the time step $(t - p + 1)$. This is also referred to as *p-step-ahead* prediction because the estimate is desired for a time instant $(t + 1)$ which is p time steps ahead of the latest measurement, which is at $(t - p + 1)$. A schematic diagram of MSP is shown in Figure 3.

Theoretically speaking, in order to perform MSP one would have to directly relate the predicted output, $\hat{y}(t + 1|t - p + 1)$, with the measurements $y(t - p + 1)$, $y(t - p)$, ..., etc. . In practice, however, MSP is performed recursively, by relating the predictions $\hat{y}(t + 1|t - p + 1)$ with the immediately prior predictions $\hat{y}(t|t - p + 1)$, $\hat{y}(t - 1|t - p + 1)$, ..., etc. . In other words, in order to perform a p -step-ahead prediction, one would perform p SSPs recursively, i.e. by utilizing the prediction in one time-step to perform the prediction in the next time-step. In fact, it can be shown that under certain simplifying assumptions, such as using an Auto-Regressive with Moving-Average (ARMA) model, the optimal mean-squared MSP error can be obtained by such a recursive procedure [71]. Furthermore, it can be argued that "good" MSP indicates that most of the deterministic dynamics of the system under study have been modeled. It is in this spirit, and by extrapolation of the aforementioned arguments

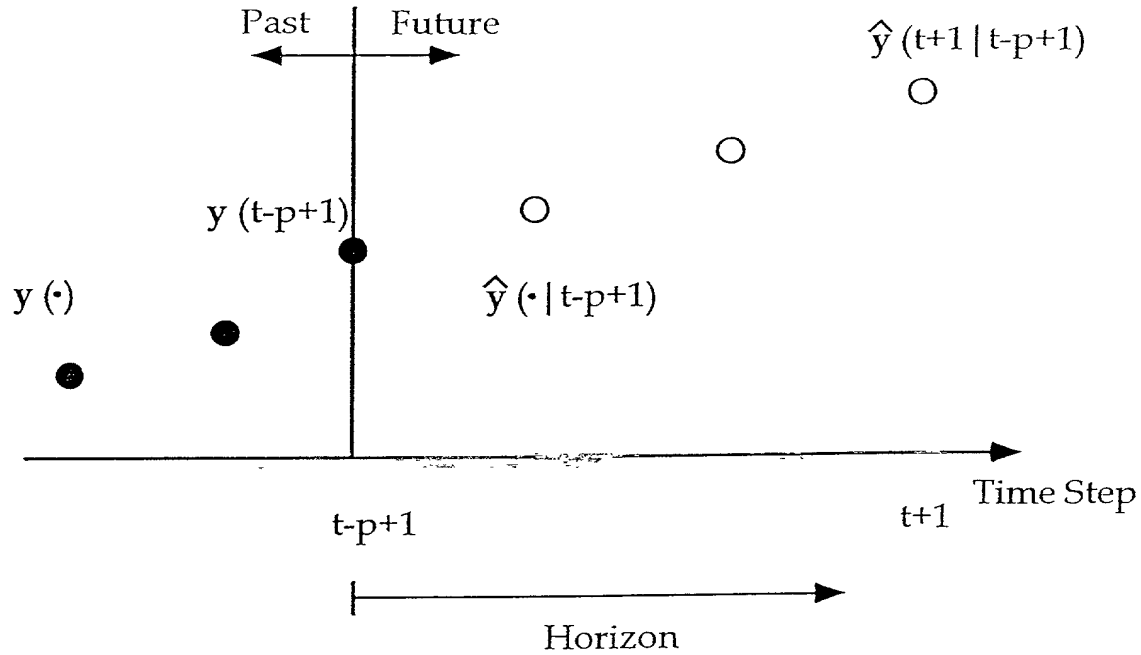


Figure 3. Multi-Step-Ahead Prediction Schematic on a p -Step Horizon.

to the nonlinear domain, that it is desired to design predictors capable of accurate MSP, as well as SSP. A block diagram of a SSP is shown in Figure 4.

In this study, and without loss of generality, it is assumed that future values of the system inputs, to the system, $u(t)$, $u(t-1)$, \dots , etc., are known exactly when used for MSP. However, if such measurements are not accurately known, then their estimates $\hat{u}(t|t-p+1)$, $\hat{u}(t-1|t-p+1)$, \dots , etc., can be used instead. In closed-loop control applications, these estimates are related to the predicted outputs via feedback.

Filtering

Filtering refers to the estimation of an unmeasurable or unmeasured variable of interest at the current time, based on measurements available up to and including the current time. It should be noted that in certain industrial set-ups, filtering is also performed on variables that are measured, as a means for an independent verification of the sensor readings.

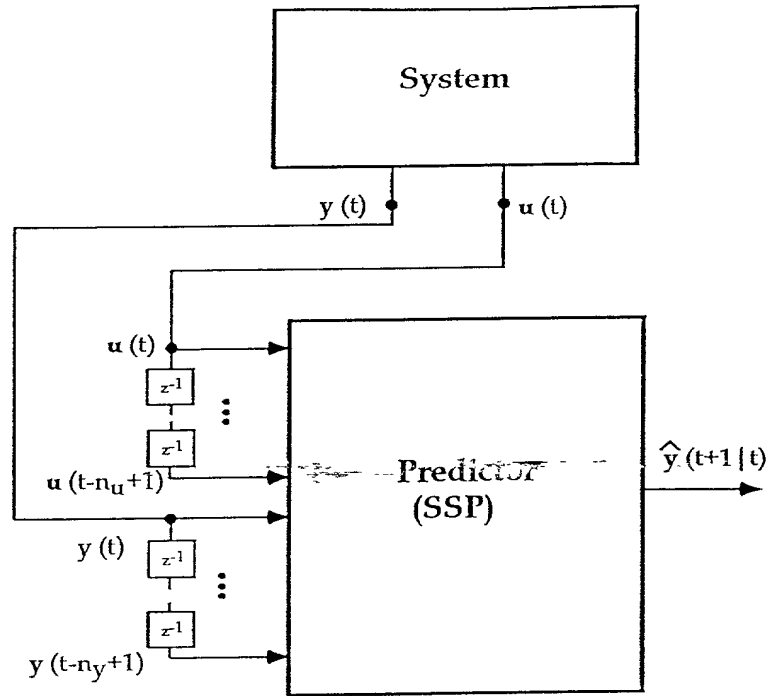


Figure 4. Block Diagram of a SSP.

The type of filtering problems considered in this study is the estimation of the unmeasurable states $x(t)$ of a system. Therefore, the filtered state value at the time instant t , $\hat{x}(t|t)$, is based on the measurements available up to and including the time instant t . A generic block diagram of a state filter is shown in Figure 5. In this figure, the scalars n_u and n_y represent the maximum number of delays used for each of the inputs and outputs, and these need not be the same for all inputs and outputs. Since the states of the system are in general not available for measurement during on-line operation, the dynamics of the state must be captured in a model of the system, either *physical* or *empirical*, for use in the estimation process. Therefore, the accuracy of the state estimate (filter output) is highly dependent on the fidelity of the available system model.

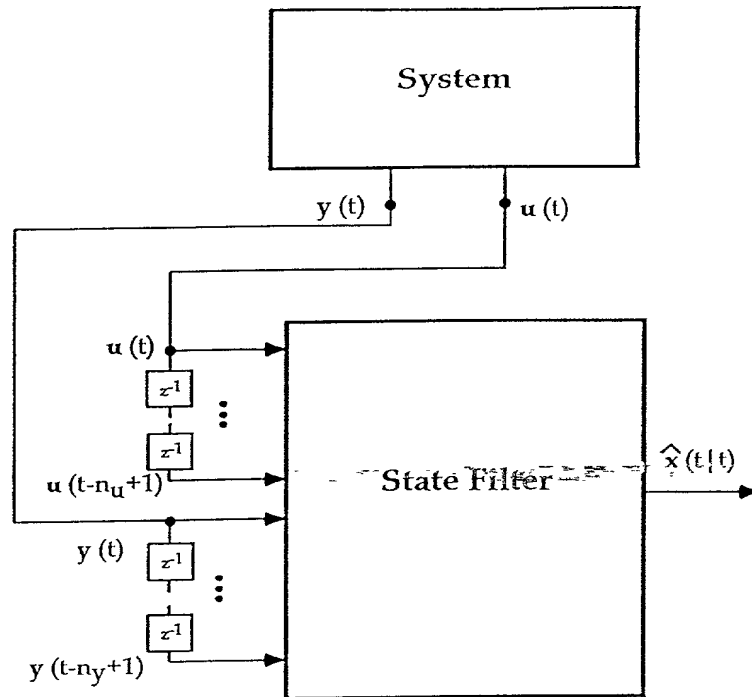


Figure 5. Block Diagram of a State Filter.

I.3.2 System Identification

The objective of SI is to infer parts of or an entire model of a system using observations. There are several steps that comprise the SI process, as follows:

- (1) *Experiment Design and Data Collection:* The purpose of this initial step is to create the experimental conditions under which the observations used in the SI process can be collected. An important aspect of this step is to ensure persistent excitation of the important dynamics to be identified.
- (2) *Model Structure Selection:* Having collected the data needed for identification, a model structure must be assumed and the specific degrees of freedom of the model structure must be specified. In this step of the process, any and all prior knowledge about the system being modeled is incorporated into the model structure selection.

- (3) *Model Parameter Estimation:* The data collected in the first step, and the model structure selected in the second step are used to set-up and solve a usually nonlinear optimization problem. The outcome of this optimization problem are the numerical values of the parameter set that completely specify the model structure selected.
- (4) *Model Validation:* The last step in the process is the validation of the identified system model. In reality, this should be called “model invalidation” step, as one attempts to disprove the model validity. If this invalidation is successful, the previous steps are repeated until one fails to invalidate the identified model.

In summary, given a finite set of observations $\{u(1), \dots, u(T)\}$ and $\{y(1), \dots, y(T)\}$, the goal of SI techniques is to determine the free parameters θ of a function $\mathcal{F}(\cdot)$ such that:

$$\hat{y}(t|t-1) = \mathcal{F}(\mathcal{U}(t); \theta), \quad (1)$$

where the “regression” vector $\mathcal{U}(t)$ is a combination of the actual system inputs $u(\cdot)$, the system outputs $y(\cdot)$, and/or the past predicted outputs $\hat{y}(\cdot|\cdot)$, etc.. Again, the predictor depicted by equation (1) can be formulated in the form of a MSP. Nevertheless, such a form necessitates more complex optimization problems for the determination of the free parameters θ .

In general, there are two methods for computing the unknown parameters θ of a model: *off-line* (or batch) and *on-line* (or recursive) methods. *Off-line* methods of SI are performed with prior measurements (data) collected from the system of interest, such as the description given in the previous paragraph. Furthermore, off-line methods can be divided into *batch* processing methods where the data is collectively used to obtain parameter estimates and *recursive* methods where the data is used one at a time to come up with parameter estimates. The latter category, however, is an *ad-hoc* approach, as the minimization performed is not in the “mean-squared” sense. *On-line* methods, by their very nature, can only be considered as recursive methods for parameter estimation.

State estimation methods, either prediction or filtering, assume that a system model is available. Thus, such methods are categorized as nonadaptive. If such a model is not available, or if the available model is inaccurate, then a model must be *identified* using SI techniques. *Adaptive estimation* refers to state estimation methods where the system model must be identified first. Therefore, adaptive estimation combines the methods of state estimation and SI. Adaptive state estimation methods, as with SI methods, can also be divided into *off-line* (batch and recursive) methods and *on-line* (recursive) methods [29].

I.4 Industrial Estimation Problems

Accurate estimates of critical parameters are of great importance in a variety of engineering applications, such as fault detection, condition monitoring, and quality control. Conventional estimation methods usually make use of an assumed linear system model. Even if a nonlinear system model is available, a linearized version of it is usually used for estimation purposes. For most complex systems, particularly complex process systems, linear models are far too inaccurate for effective state estimation. This is the case, even though such models are sufficient for control applications. Furthermore, numerous industrial systems of interest are far too complex to model from first-principles.

Many process and manufacturing industries use physical models for design purposes. However, these models, mostly steady-state, are often too complex and cumbersome to be effectively used for on-line applications. Physical models also suffer from the difficulties involved in incorporating poorly understood physical effects, such as wear and tear effects. Furthermore, physical models suffer from difficulties in incorporating uncertainties in parameters due to plant “drifts”. These drifts could be the results of aging, corrosion and/or material defects, etc. . Empirical models, on the other hand, are well-suited for on-line applications as they are computationally less cumbersome than physical models. More importantly, however, on-line applications

require the means to occasionally or continuously update the available models using sensor information in order to compensate for the ever present modeling uncertainties and changes in system behavior. This important and usually critical feature, which is easily incorporated into empirical and semi-empirical models, is exceedingly difficult to incorporate into physical models.

Although most physical models of complex process systems are not suitable for on-line applications, such models are still required in estimation problems using empirical methods. The states, for which an estimate is required, frequently cannot be measured on-line (or even off-line in some cases). The correlation (dynamic relationship) between the states and the measured quantities, which has to be incorporated in the empirical model design, must come from the physical model of the system. Alternatively, this information can be obtained from off-line, and occasionally expensive experiments, if at all such experiments can be performed. Therefore, it is desirable to develop state estimation methods that use physical models in combination with empirical models, if the use of physical models for on-line applications is a viable option. In this study such methods are called “hybrid methods”. If hybrid methods are not feasible, then estimation methods using only empirical models become the only alternative.

Estimation methods, in general, and NN-based adaptive estimation methods, in particular, have many industrial applications, such as in *inferential sensing*, *inferential control*, *dynamic data reconciliation* (or rectification), *model predictive control* (MPC), *adaptive control*, *condition monitoring and condition assessment*, and *fault diagnosis and prognosis*. All of the aforementioned problems depend in one way or another on the effective solution of an, usually adaptive, estimation problem.

Inferential sensing is another term used for state filtering and has many applications in process systems, where the state to be estimated is not measured [81]. *Inferential control* refers to the control of a system output which is measured using its “inferred”(estimated) value from the system outputs that are physically measured. This type of control is important also in a number of process systems, such as those

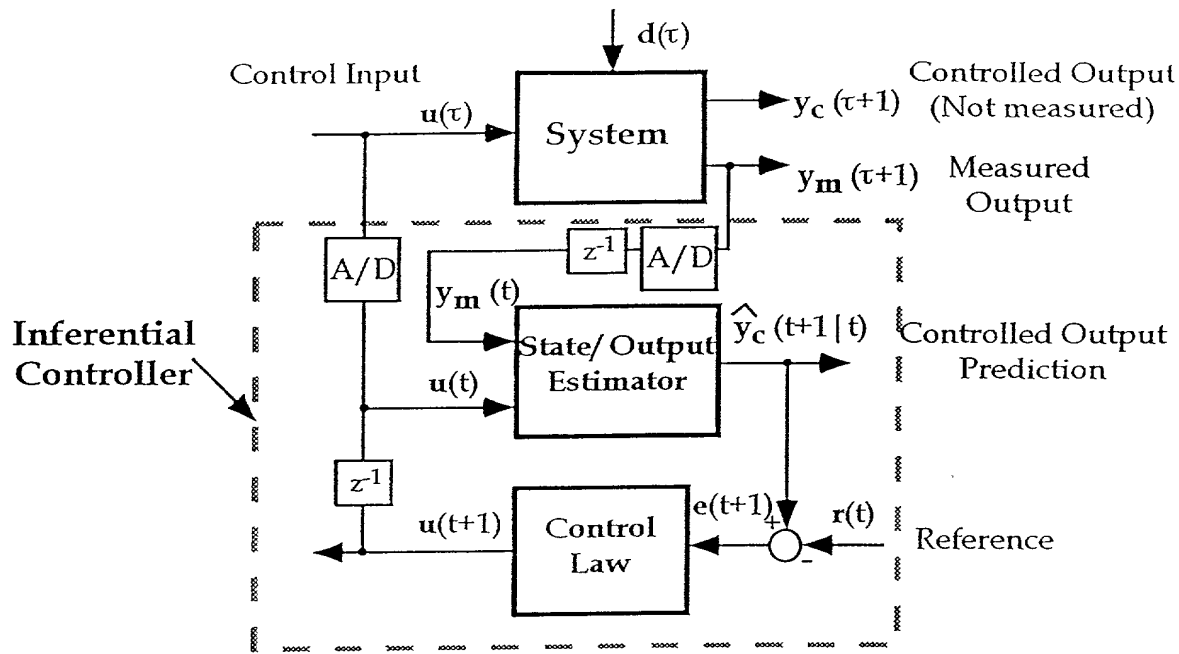


Figure 6. Block Diagram of an Inferential Control Scheme.

found in the manufacturing and chemical industries [47]. A block diagram of an inferential controller is shown in Figure 6.

Dynamic data reconciliation or rectification is also another term used for filtering output measurements. This is useful in sensor validation methods or in processes with very noisy measurements. MPC is a process control method that explicitly uses a model of the process under consideration to predict *future* values of the controlled process output. This, in turn, is used to determine only the *current* values of the control inputs [24]. This type of control method is also referred to as *anticipatory* control.

In *adaptive control*, an algorithm is formulated to regulate the outputs to fixed set-points or set-points that change in a predetermined manner during normal operation. In adaptive control schemes, unlike inferential control schemes, the controlled

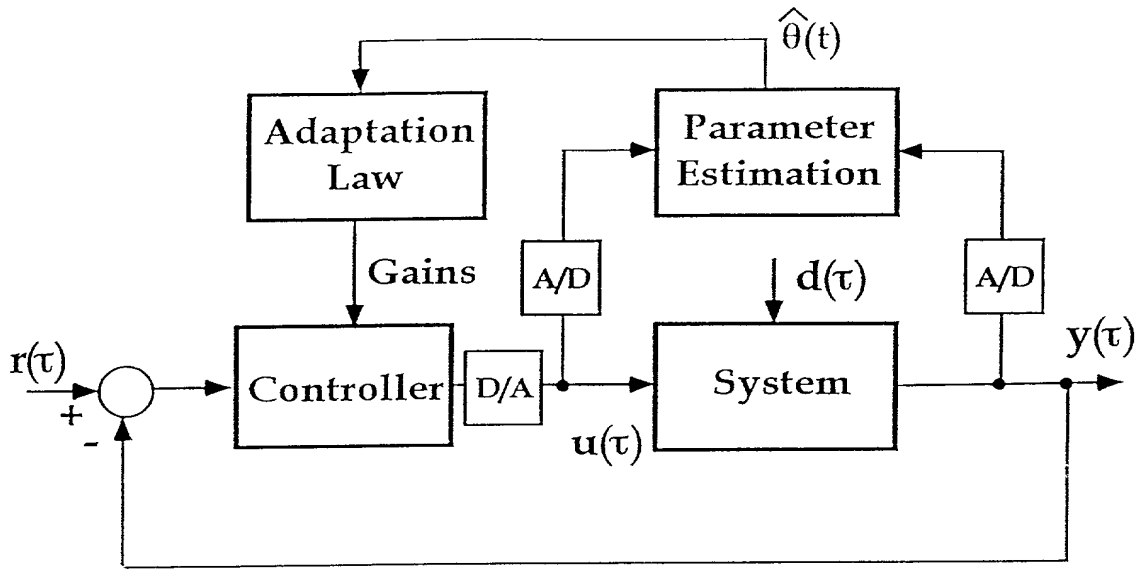


Figure 7. Block Diagram of an Adaptive Control Scheme.

output is assumed measured. The gains of the controller are computed based on parameter values that are estimated using measurements from the system. A diagram of a typical adaptive control scheme is shown in Figure 7.

Frequently, these algorithms are dependent upon having accurate state estimates before a control law can be calculated. If the state estimate is highly erroneous, then control of the system will subsequently be quite poor. Thus, accurate state estimation methods become essential in many industrial control problems.

Another important application area where accurate estimation methods are necessary is in condition monitoring, fault diagnosis and prognosis [6],[92],[94],[95]. Enhanced reliability in dynamic systems is traditionally achieved through the use of hardware redundancy. Recently, new approaches have been developed which seek to eliminate some or most of the redundant hardware. These new approaches are based on the idea that two (or more) dissimilar sensors measuring different variables can be used in a comparison scheme to detect incipient system failures. These functionally redundant schemes are basically signal processing techniques employing state and/or

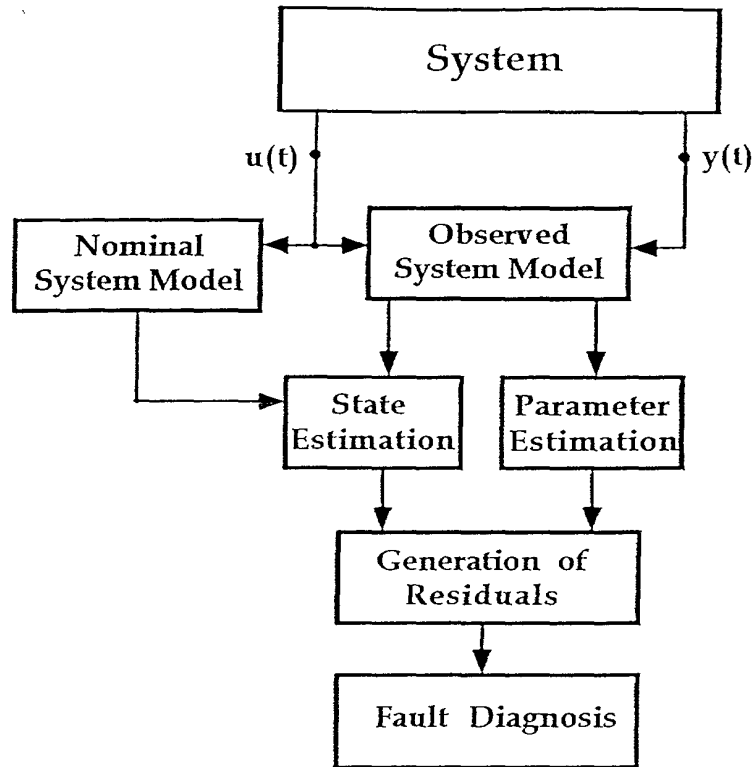


Figure 8. Block Diagram of a Fault Diagnosis Scheme Using Estimation.

parameter estimation methods, which can be implemented in software using off-the-shelf digital computers [59]. A block diagram showing such a fault diagnosis scheme is shown in Figure 8.

Thus, fault diagnosis schemes are designed based on the assumption that either the dynamic states of the system under study are known, e.g. by estimation, to a certain precision, or that it is possible to determine the values of certain physical parameters by on-line identification techniques.

I.5 The Role of Neural Networks in Estimation

From the preceding discussion it is apparent that a large class of industrial estimation problems are reliant on high-fidelity nonlinear dynamic system models.

This section will address the role of NNs in nonlinear estimation and, in a broader context, in nonlinear system modeling. Furthermore, the potential advantages of NNs over traditional nonlinear modeling approaches will be explained.

For the sake of simplicity, and without loss of generality, assume that it is desired to model a static (memory-less) system that has only deterministic nonlinearities, i.e. no stochastic components are present. The arguments presented here are equally applicable to nonlinear dynamic systems, and furthermore to stochastic (nonlinear) dynamic systems. However, the mathematics become exceedingly complex, as it will be witnessed by later chapters in this dissertation.

Further, assume that a model of this static system can be expressed by the function $f(\cdot)$, expanded using the Taylor Series as follows:

$$y = f(x) = \sum_{i=1}^{\infty} a_i x^i = a_1 x + a_2 x^2 + \cdots + a_n x^n + \cdots, \quad (2)$$

where x is the independent variable, system input, and y is the dependent variable system output. The *linear* treatment of nonlinearities is to keep the first term of the expansion (2) and ignore the higher order terms. The resulting “linear” approximation is represented by the following equation:

$$y = f(x) \approx a_1 x. \quad (3)$$

This approximation may be adequate for certain applications. However, in general, a linear model of a nonlinear system might be too constraining and inaccurate, especially for estimation purposes. This is even more so in numerous industrial applications. Therefore, in some instances it is not only desired but it is necessary to model system nonlinearities.

The traditional analytical treatment of nonlinearities is to make increasingly restrictive assumptions about $f(\cdot)$, such as boundedness, smoothness, etc., and then proceed in addressing and solving the problems with arguments based on mathematical analysis. As a result of the need to proceed within the limitations provided by the analytical tools of mathematics, the assumptions made severely restrict the classes of nonlinearities that can be dealt with. In fact these approaches preclude a large

number, if not most, of the real-world systems encountered in engineering practice. Nonlinear models based on such approaches are usually used in analytic methods to estimation and control, e.g. Kalman Filters, Lyapunov methods, etc, [28],[70].

However, from a more practical standpoint, system nonlinearities may not be in any known form. Even if they were, they may not be in a known analytical form. To overcome these limitations, it might be necessary to resort to “approximation”-based approaches in modeling nonlinearities. In this approach, system nonlinearities are *inferred* (approximated or estimated) from system measurements. Therefore, the “approximation” treatment of nonlinearities would involve truncation of equation (2) as follows:

$$y = f(x) \approx \sum_{i=1}^n a_i x^i \approx a_1 x + a_2 x^2 + \cdots + a_n x^n, \quad (4)$$

and computing the expansion coefficients a_1, a_2, \dots, a_n using the measured system data. It is pertinent to note that the true “power” of such empirical approximations for the treatment of the system nonlinearities is entirely dependent on the expansion (basis functions) used.

The traditional methods of modeling nonlinearities via approximations, such as the Volterra series expansions, use polynomial expansions. The practical implications of such expansions is that each term of the expansion (i.e. the terms x , x^2 , x^3 , etc.) is structurally different. Consequently, complications arise when attempting to systematically determine the expansion coefficients for increasingly more accurate expansions. Specifically, in the case of the Volterra series expansions, the number of expansion coefficients to be determined grows prohibitively large for a modest number of expansion terms [7],[16].

NN expansions, on the other hand, can be expressed by the following approximation:

$$y = f(x) \approx \sum_{i=1}^n a_i g_i(x) = a_1 g_1(x) + a_2 g_2(x) + \cdots + a_n g_n(x), \quad (5)$$

where $g_i(x)$ could be a gaussian, a sigmoidal, or any other “squashing” function, instead of a polynomial. The term “squashing” function includes all functions with a

finite active region, i.e. non-zero first derivative only in a finite region. Comparing equations (4) and (5), it is observed that each term of the expansion (5) is structurally similar. The implication of such an expansion is that for increasingly more accurate expansions, a less complex and a computationally less intensive approach is possible for determining the expansion coefficients, as compared to the polynomial approximation methods. Furthermore, the use of the “squashing” function in the expansion of (5), whether local or global, is the main reason that the NN approaches approximate a wider class of functions than conventional polynomial approximation methods [4],[5]. The basic argument of using NNs as “squashing” basis functions for approximating system nonlinearities has biological motivations [86],[88], and it becomes the main motivation for using these expansions in dynamic estimation and control problems. Estimation methods based on NNs appear well-suited for such dynamic estimation and control problems. Models resulting from these expansions can be designed in a *nonparametric* manner. This is because even though NNs, in itself, are parametric, the ease of derivation of NN models render them nonparametric.

I.6 Literature Review

I.6.1 Estimation

Gauss [25] is the first person credited for having mathematically formalized the problem of estimation, while attempting to solve the problem of estimating the orbit of the asteroid Ceres via the least-squares technique (which he also developed in 1795). Since then, the area of estimation was developed primarily on probabilistic arguments. In this development, the formulation and solution of the linear estimation problem were based on assumptions made about the probability distribution function (PDF) of the measurements and quantity to be estimated. In addition, only input-output models of the systems under consideration were used. Some examples of these methods are Bayesian estimation, maximum-likelihood estimation, Wiener filtering etc. [1],[2]. These methods are considered as the *classical* methods of linear estimation, though the theory of Wiener filtering was later extended to nonlinear systems

[89]. The main disadvantage of these methods were the assumptions placed on the statistics of the measurement data, which were not often valid in many applications. Furthermore, the solutions to estimation problems, using the classical methods, were too cumbersome and difficult to implement for most on-line applications.

In the 1960s, Kalman and coworkers [39] solved the problem of estimation in linear systems with a recursive least-squares type of solution. This solution was, in a sense, a modern extension of Gauss' original least-squares solution to the state estimation problem. This paper formulated the so-called Kalman Filter (KF) which has since found numerous industrial applications. This seminal work of Kalman was significant for several reasons [72],[76], among them:

- (1) The recursive least-squares approach, like the least-squares approach formulated by Gauss, does not (in general) require any statistical concepts or assumptions in its formulation. This is in sharp contrast to the classical estimation methods which made assumptions regarding the statistics of the underlying processes. In practice, however, even the KF algorithm makes certain mild assumptions about the statistics of the measurement and process noise. These assumptions are usually violated in practice and often seriously affect the performance of the KF.
- (2) An explicit model, in a *state-space* form, of the system under study was used in the solution of the estimation problem. The state-space approach uses differential or difference equations, which has now become the conventional approach to describing dynamical systems.
- (3) The KF equations are extremely easy to implement on a digital computer. In contrast, the classical filter equations, such as the Weiner-Kolmogorov filter etc., are very difficult to implement because of their non-recursive nature.

Furthermore, all the results obtained using classical estimation techniques can be obtained from the KF approach under the same assumptions [1]. In fact, the solutions to the state estimation problem can be classified into two types [38]: the *Direct Kalman Gain* approach, and the *Indirect Noise Covariance* approach. In the

direct approach to state estimation, the KF gain is computed directly from the measurement data [43]. In the indirect approach, the statistics of the measurement and process noises are computed first, and then these are used to compute the KF gain [28].

In the case of solving the estimation problem for nonlinear systems, the so-called Extended Kalman Filter (EKF) was formulated. Here the model of the nonlinear system is linearized about a fixed point or a trajectory. The standard KF equations are then applied to the linearized model of the system [66]. This method has been found to work well for some systems where the nonlinearities are mild. However, in general, this method suffers from problems of divergence of the estimate over time. As an alternative to the EKF for solving nonlinear estimation problems, researchers have been looking into approximating the probability density functions of the measurement and process noise instead of approximating (linearizing) the nonlinear system model [77]. However, this method has not been shown to be very promising in process systems where assumptions about the statistics of the variables involved cannot be made with any degree of accuracy.

Since the 1960s, the majority of applications of estimation in general, and state estimation in particular, have been towards the design of more robust control systems for dynamical systems with significant modeling uncertainty. Therefore, estimation problems have always been dealt with as a means to designing improved control systems. Moreover, most of the available methods are linear and have limited scope in application to dynamical systems with significant nonlinearities. An emerging area where estimation methods are deemed important is in condition monitoring and fault detection. Isermann has applied model-based linear estimation techniques to detect faults in electro-mechanical systems [35],[36],[37]. Frank [23] has developed banks of observers where the output of each observer indicates if a specific fault has occurred or not. Watanabe and coworkers [83],[84] have applied EKFs to the problem of estimating parameters of a chemical process system for fault detection. Clark [17], Eckert [20], Kitamura [40], and Tylee [82] have developed linear estimation schemes to detect faults in nuclear power plant components. More recently, researchers have

initialized work on the applications of NNs to general nonlinear filtering problems with potentially significant commercial benefits [46].

I.6.2 System Identification

SI can be categorized into two main categories: Linear system identification and nonlinear SI. However, most of the available literature on SI is concentrated in the linear domain. Linear systems are usually idealizations of nonlinear systems encountered in real-world applications. If system nonlinearities are negligible, then application of linear SI leads to good results. For a thorough review of linear SI, the reader is referred to a number of recently available excellent books [43],[44],[71].

The literature for applied and computationally oriented nonlinear SI is quite scarce, though there have been numerous theoretical studies dating back to the early 1900s. Traditionally, functional series method, such as the Volterra and Wiener series, have been used for the identification of nonlinear systems [7], [31]. It is well known that these structures can describe a number of nonlinear systems, however, they are not very convenient for practical, especially on-line, use. Billings and his colleagues were the first to report the polynomial Nonlinear Autoregressive with eXogeneous input (NARX) model structure [8],[10],[11],[12]. The polynomial Nonlinear Autoregressive and Moving Average with eXogeneous input (NARMAX) nonlinear model structure was also introduced by the same researchers. They have worked extensively in the area of input-output nonlinear SI using a number of methods. The polynomial NARX appears to be one of the most promising structures for representing nonlinear systems. Later in this dissertation both NARX and NARMAX model structures are presented in more detail.

I.6.3 Neural Networks

A NN is a biologically inspired computational paradigm which has multiple interconnected processing elements grouped into layers as linear arrays. Each processing element is characterized by a simple nonlinear operator. The processing power

of a NN is a combination of its specific processing element structure and its network topology. A special class of NNs can be constructed using functional neurons called perceptrons [67]. Trained by adaptation using a cost criterion, NNs are believed to be good at interpolation and extrapolation. The path between the nodes (or the counterparts of the biological dendrites) are modeled by the NN links. The connections between neurons in a network fundamentally determine the behavior of the network and how that behavior can change with time. For this reason, the field today known as NNs was originally called *Connectionism*.

Connectionism was born during the middle decades of this century. In the 1940s, Warren McCulloch and Walter Pitts explored the computational capabilities of networks made-up of model neurons with a very simple design. A McCulloch-Pitts model neuron fires if the sum of its excitatory inputs exceeds its threshold, so long as it receives no inhibitory input [60]. Networks of this type seemed appropriate for modeling not only symbolic logic, but also perception and behavior. At the same time during the 1940s, Wiener and his co-workers laid the foundations of the field known as cybernetics, or the control and communications in man and machine, which was a prelude to the field of modern artificial intelligence [88]. In the 1950s, Rosenblatt [67] introduced a mathematical analysis of the behavior of a class of network models called perceptrons. The perceptron exhibited some learning and generalization capabilities. After going through many other stages including a phase of significant lack in progress, NNs or, more precisely, artificial neural networks (ANNs), regained momentum through the work of Hopfield, Rumelhart, Williams, and Werbos. The latter three were instrumental in advancing the so-called Backpropagation (BP) algorithm throughout the later half of the 1980s and in the early 1990s [68],[86],[87].

Many NN architectures and learning algorithms have been suggested by different researchers and they are used to solve a number of problems of practical importance. For a review of a wide range of NN architectures and learning algorithms, the reader is referred to two recent texts [30],[33]. Nevertheless, for estimation applications in dynamic systems, only a few of these architectures appear promising. One of the most promising NN model structures is the Feedforward Multilayer Perceptron (FMLP)

neural network [49],[50]. FMLP networks are characterized by sets of nonlinear algebraic equations, one for each node of the network. Cybenko [18] proved that a three-layer FMLP network is capable of approximating any function with an arbitrary amount of approximation error. The number of nodes, n , of the FMLP network that are required to do this was not specified and must be determined in an ad-hoc manner. Barron [4],[5] quantified this result and showed that the mean integrated squared error between the FMLP network and the target function, $f(\cdot)$, was bounded by a value which was the sum of two terms: the bound on the approximation error and the bound on the estimation error. This is expressed as:

$$\frac{1}{N} \sum_{i=1}^N \left[f(X_i) - \hat{f}_{NN}(X_i) \right]^2 \leq \frac{C_f^2}{n} + \frac{n d}{N} \log N, \quad (6)$$

where n is the number of nodes, d is the input dimension of the function, N is the number of training observations, C_f^2 is the first absolute moment of the Fourier magnitude distribution of $f(\cdot)$, $\hat{f}_{NN}(\cdot)$ is the estimated NN function, and X_i is the i -th training observation. The first term on the right-hand-side of equation (6) is the bound on the function approximation error and it is *inversely proportional* to the number of nodes, n . The second term on the right-hand-side (RHS) of equation (6) is the bound on the NN estimation error and it is *directly* proportional to n as well as the input dimension of the function, d . Barron [4] has shown that while the bound on the approximation error of NNs is proportional to $(1/n)$, the bound with conventional approximation functions, such as polynomials, splines, and trigonometric expansions, is proportional to $(1/n)^{2/d}$. Therefore, the approximation error with NNs is lower than with conventional approximation methods when $d \geq 3$. This improved approximation error becomes increasingly more apparent as the dimension of the input space increases. This superior performance of NNs is ascribed to the use of the sigmoidal “squashing” functions.

The aforementioned developments address problems and issues of a “static” nature, i.e. those found in memory-less systems. If temporal processing is the subject of investigation, then feedback must be incorporated in the NN architectures as well

as in the learning algorithms. There are several ways to introduce feedback, the simplest one being augmenting the FMLP with a NARX-type (global or external) feedback. The resulting NN is a dynamic feedforward NN and it has been studied by many in the literature [49].

Another way of introducing feedback to NNs, is to allow local (or internal) feedback. The resulting NN architecture is called recurrent. Recurrent networks have been introduced in 1982 by Hopfield [34], though their use in temporal processing has only been recently investigated. The Recurrent Multilayer Perceptron (RMLP) is such a recurrent architecture, proposed in 1990 [56]. As demonstrated in previous research studies, for complex systems, such as the process systems treated in this research, the RMLP is by far a more appropriate architecture to adopt than the FMLP [16],[55],[56],[57],[65]. If global feedback is introduced to a recurrent NN, such as a RMLP, the resulting architecture is called *dynamic* recurrent NN, and this approach has become the subject of very recent investigations [80],[65].

The BP learning algorithm, which is a parameter estimation technique used in training FMLPs with a gradient descent method, has played a significant role in the resurgence of interest in NNs [68]. Several techniques and algorithms have been developed for training various forms of recurrent and dynamic recurrent NNs. Williams and Zipser [90], and Williams and Peng [91] have developed an algorithm for fully connected recurrent networks, the so-called dynamic backpropagation. Narendra and Parthasarathy [51] have discussed the dynamic backpropagation learning algorithm, and its application for the optimization of FMLP NN parameters. They emphasized the diagrammatic representation of the system which generates the gradient of the performance function. Additionally, Werbos [85], Pineda [61],[62], and Parlos and coworkers [57] have proposed algorithms for training recurrent and dynamic recurrent NNs.

NNs have been applied in a variety of areas ranging from pattern recognition to adaptive control. Several researchers have applied NNs for the identification of nonlinear systems [9],[14],[57],[58]. Lo [46] has proposed using recurrent NNs for filtering on some stochastic nonlinear systems. Williams [93] and Puskorius [64] have used an

EKF to determine the weights of recurrent neural networks in control applications. DeCruyenaere and coworkers [19] have compared the performance of the KF and recurrent NNs in the estimation of states for a number of different problems. These researchers have reported that in most cases the performance of the NNs appears better than that of the KF. Lainiotis [41],[42] has used recurrent NNs for estimating the states of some simple nonlinear dynamical systems. Portman and collaborators [63] have applied a combination of a physical model along with a NN model to control a rolling steel mill. In this application, the NN was used to estimate some process parameters which were then fed to the physical model. In this way, the NN was used to enhance the performance of the physical model with accurate estimation of some process parameters. Tsaptsinos and coworkers [53] have used NNs state estimation of a chemical fermentation process and compared the results with a KF approach. They have reported that the NN performed better.

As witnessed by the literature review of the last paragraph, most NN application to estimation of dynamic systems have been limited to SI and the resulting SSP problems. There have been less than half a dozen published studies of attempts to use NNs for filtering applications and one on MSP. Of the published work on filtering applications, most have dealt with specific problems of such simplicity that even traditional filtering methods would address equally well. Therefore, the stage has been set and the motivation presented for applying NNs to real-world complex (adaptive) filtering problems.

I.7 Research Contributions

The main contribution of this research study is to develop a method for state filtering in complex process systems using recurrent NNs. The proposed state filtering method can be utilized in a non-adaptive or in an adaptive manner, as dictated by the needs of the specific application. The key distinction between the proposed method and other conventional state filtering methods is the minimal, if any, assumptions placed upon the assumed process model. The main assumption imposed upon the

utilized process models stems from the approximation limitations of the utilized NNs. As a result, the proposed method is nonparametric and nonlinear in nature, and, it has innumerable applications in the areas of process control, condition monitoring, and fault diagnosis.

In particular, as a result of the research described in this dissertation, the following two specific contributions have been made in the area of state filtering:

- (1) A new state filtering method is proposed based on the fundamental principles of Kalman Filters and least-squares estimation. The new method utilizes recurrent NNs, and it is nonparametric and nonlinear in nature. It can be formulated for use in a non-adaptive and in an adaptive manner.
- (2) The effectiveness of the developed state filtering method is demonstrated on a DC Motor-Pump and a U-Tube Steam Generator (UTSG). The former is a moderately complex process system, while the latter is a highly complex process system. The performance of the developed state filtering method is extensively tested, and it is demonstrated that it has the accuracy and practical applicability required by many industrial filtering problems.

I.8 Organization of Dissertation

In Chapter II, an overview of linear models and the associated estimation, prediction and state filtering, methods are presented. The standard KF is introduced. Nonadaptive methods for linear estimation, linear SI methods and adaptive linear estimation methods are briefly reviewed.

In Chapter III, an overview of nonlinear prediction methods is presented. Following the introduction of a general framework for nonlinear predictors, conventional expansions are introduced. System models and the associated prediction methods using conventional nonlinear expansions are described. Nonadaptive methods of nonlinear prediction, nonlinear SI methods, and adaptive nonlinear estimation methods are presented. This is followed by NN expansions. The FMLP and RMLP are introduced as alternate functional expansions for nonlinear process system models.

Prediction methods using NNs are described. Nonadaptive NN prediction methods, learning algorithms, and adaptive NN prediction methods are then presented.

In Chapter IV, an overview of nonlinear state filtering methods is presented, along with the main contributions of this dissertation. The conventional nonlinear state filtering method, the EKF, is presented. NN state filtering methods, the subject of this dissertation, are then developed. Nonadaptive NN state filtering methods, learning methods, and adaptive NN prediction methods are presented.

In Chapters V, VI, and VII the proposed state filtering methods are applied to an artificial Two-Input-Two-Output (2I2O) system, a DC Motor-Pump system, and a UTSG process system, respectively. Extensive simulation tests are conducted on each of these systems to evaluate the performance of the developed state filtering methods on these increasingly more complex process systems.

A summary of this dissertation, the conclusions reached from this research, and directions for further future research are given in Chapter VIII.

AN OVERVIEW OF LINEAR ESTIMATION METHODS

II.1 Introduction

The estimation problems that are considered in this study are that of *prediction* and *filtering*. *Prediction* refers to the estimation of a variable of interest at a future time, given measurement data up to and including the present time. *Filtering* refers to the estimation of a variable of interest at the present time, given measurement data up to and including the present time. This chapter provides an overview of some methods used in *linear* estimation. This implies that the system model under study is assumed linear.

As mentioned earlier, the majority of currently addressed estimation problems use linear models and linear methods. This is mostly because of the abundance of linear analytic methods of solution. Furthermore, the majority of linear estimation problems use the Linear Least-Squares (LLS) method of solution or variations thereof. The LLS methods, as applied to estimation, finds the estimate that minimizes the mean-square of the estimation error. One is guaranteed to find the minimum of the estimation error, as long as linear models are used [29]. For this reason, the LLS methods and its variants have been very successful. One variation of the LLS method that has been proved to be more practical for on-line applications is the Recursive Least-Square (RLS) method. The RLS method provides an estimate as soon as each observation is made.

Linear input-output models that are frequently used are the Auto-Regressive with eXogeneous inputs (ARX) and Auto-Regressive Moving Average with eXogeneous inputs (ARMAX) models. The use of linear state-space models was popularized in the 1960s, and this is a more natural way of representing dynamical systems. Furthermore, several powerful analysis methods exist for state-space models that are too

cumbersome or are lacking for input-output type models. The Kalman Filter (KF) is a method for estimating the outputs and states of a system recursively, and it explicitly utilizes the state-space system model. When a system model is unknown, then system identification (SI) techniques must be applied to determine the parameters of these models. This situation arises in systems that experience changing operating conditions or system component attrition and/or defects etc. . In this case, the existing linear models used in estimation are too inaccurate and must be identified again. In SI techniques, the measurement data is used to determine the parameters of the linear models. If SI is followed by some type of estimation, then this is termed as adaptive estimation.

This chapter is organized as follows: Section II.2 describes linear, static and dynamic, system models. Among the dynamic models described are the ARX, the ARMAX, and the state-space models. Section II.3 describes nonadaptive estimation, prediction and state filtering, based on linear static and dynamic process system models. Section II.4 discusses SI methods with linear input-output models and state-space models. Section II.5 briefly mentions adaptive methods of linear estimation. Section II.6 summarizes the chapter.

II.2 System Models

Linear estimation algorithms assume that the system under investigation can be adequately represented by linear models. Linear models are of two types: static models and dynamic models. Dynamic linear models can be further classified into two basic types: *Linear Input-Output* models which, as the name implies, model the dynamic relationship between the system inputs and outputs in the form of a linear regression, and, *Linear State-Space* models which model the dynamic relationship between the system inputs and outputs through intermediary descriptor variables referred to as the states. These models are discussed in some detail in the following sections.

II.2.1 Static Models

A linear, static system model expressing the relationship between the model output, $\mathbf{y}(t)$, and the model state, \mathbf{x} , can be represented by the following equation:

$$\mathbf{y}(t) = \mathbf{H}\mathbf{x} + \mathbf{v}(t), \quad (7)$$

where $\mathbf{y}(t)$ is an $N \times 1$ dimensional vector, \mathbf{x} is an $N \times 1$ vector, \mathbf{H} is an $N \times N$ matrix and $\mathbf{v}(t)$ is an $N \times 1$ noise vector, i.e. a vector stochastic process.

II.2.2 Dynamic Models

In this section, models relating dynamic relations between the system input and output variables are discussed. Dynamic systems are characterized as possessing memory. This implies that the output of the system is dependent on present and past values of the inputs and outputs. Dynamic models can be classified into two categories: input-output models, and state-space models

II.2.2.1 Input-Output Models

One of the most commonly used linear input-output model is the ARX model. A general ARX model for a multiple-input multiple-output (MIMO) system is represented by the following equation :

$$\begin{aligned} \mathbf{y}(t+1) = & \mathbf{A}_1 \mathbf{y}(t) + \cdots + \mathbf{A}_{n_y} \mathbf{y}(t - n_y + 1) \\ & + \mathbf{B}_1 \mathbf{u}(t) + \cdots + \mathbf{B}_{n_u} \mathbf{u}(t - n_u + 1) + \mathbf{e}(t), \end{aligned} \quad (8)$$

where $\mathbf{y}(t)$ is the output of the ARX model, n_y is the number of past outputs (lag terms) used in the model, $\mathbf{u}(t)$ is the input to the ARX model, n_u is the number of past inputs (lag terms) used in the model, and $\mathbf{e}(t)$ is the assumed noise term. Usually, the vector noise term, $\mathbf{e}(t)$, is assumed to be a zero-mean, white, Gaussian process. Furthermore, the coefficients of the ARX model, the elements of the matrices \mathbf{A}_i and \mathbf{B}_j for $i = 1, 2, \dots, n_y$ and $j = 1, 2, \dots, n_u$ are assumed to be known, or computable in some way (this is the subject of SI techniques).

The general single-input single-output (SISO) ARX model is represented by the following equation:

$$\begin{aligned} y(t+1) = & a_1 y(t) + \cdots + a_{n_y} y(t - n_y + 1) \\ & + b_1 u(t) + \cdots + b_{n_u} u(t - n_u + 1) + e(t), \end{aligned} \quad (9)$$

where $y(t)$ is the output of the SISO ARX model, n_y is the number of past outputs (lag terms) used in the model, $u(t)$ is the input to the ARX model, n_u is the number of past inputs (lag terms) used in the model, and $e(t)$ is the assumed noise term. Usually, the noise term, $e(t)$, is assumed to be a zero-mean, white, Gaussian process. Furthermore, the coefficients of the ARX model, a_1, \dots, a_{n_y} and b_1, \dots, b_{n_u} are assumed to be known, or computable in some way (this is the subject of SI techniques).

A more general input-output model is the ARMAX model. For a MIMO system, an ARMAX model is represented by the following equation:

$$\begin{aligned} \mathbf{y}(t+1) = & \mathbf{A}_1 \mathbf{y}(t) + \cdots + \mathbf{A}_{n_y} \mathbf{y}(t - n_y + 1) \\ & + \mathbf{B}_1 \mathbf{u}(t) + \cdots + \mathbf{B}_{n_u} \mathbf{u}(t - n_u + 1) \\ & + \mathbf{e}(t) + \mathbf{C}_1 \mathbf{e}(t-1) + \cdots + \mathbf{C}_{n_e} \mathbf{e}(t - n_e + 1), \end{aligned} \quad (10)$$

where n_e are the number of the past noise terms (lag terms), and the elements of the matrix \mathbf{C}_k for $k = 1, 2, \dots, n_e$ are assumed to be known. The other variables in equation (10) are the same as in the MIMO ARX model.

The SISO ARMAX model is represented by the following equation:

$$\begin{aligned} y(t+1) = & a_1 y(t) + \cdots + a_{n_y} y(t - n_y + 1) \\ & + b_1 u(t) + \cdots + b_{n_u} u(t - n_u + 1) \\ & + e(t) + c_1 e(t-1) + \cdots + c_{n_e} e(t - n_e), \end{aligned} \quad (11)$$

where n_e is the number of the past noise terms (lag terms), and where the other variables are the same as in the ARX model.

II.2.2.2 State-Space Models

State-space models represent the relationships between the inputs, noise and output signals of a system as a set of first-order differential or difference equations, using an auxiliary *state* vector, $\mathbf{x}(t)$. This description of linear dynamical systems is sometimes more useful than the input-output descriptions, because it incorporates the physical principles of the system into the model.

A discrete-time formulation of the state-space description of a linear system is represented by :

$$\mathbf{x}(t+1) = \mathbf{A} \mathbf{x}(t) + \mathbf{B} \mathbf{u}(t) + \mathbf{w}(t), \quad (12)$$

$$\mathbf{y}(t) = \mathbf{C} \mathbf{x}(t) + \mathbf{v}(t), \quad (13)$$

where $\mathbf{y}(t)$ is an $n \times 1$ vector and represents the outputs of the state-space model; $\mathbf{u}(t)$ is an $m \times 1$ vector and represents the inputs to the model; $\mathbf{x}(t)$ is an $l \times 1$ vector and represents the states of the model; \mathbf{A} , \mathbf{B} and \mathbf{C} are matrices of appropriate dimensions ($l \times l$, $l \times m$, and $n \times l$, respectively); $\mathbf{w}(t)$, the system noise, is an $l \times 1$ vector; and $\mathbf{v}(t)$, the measurement noise, is an $n \times 1$ vector. The system noise, $\mathbf{w}(t)$, and the measurement noise, $\mathbf{v}(t)$, are usually assumed to be zero-mean, white Gaussian processes. The model presented by equations (12) and (13) is called the “noise” representation of the state-space.

II.3 Nonadaptive Methods

In this section, it is assumed that a linear system model is available for use in estimation, specifically in prediction and/or filtering. These linear estimation methods are discussed in the following subsections.

II.3.1 Static Systems

II.3.1.1 Least-Squares Method

Assume that N measurements, \mathbf{y} , where $\mathbf{y} = [y_1, \dots, y_N]$, from a static system are related to the constant state vector, \mathbf{x} , by the equation (7). The estimated system output is represented by :

$$\hat{\mathbf{y}} = \mathbf{H} \hat{\mathbf{x}}, \quad (14)$$

where $\hat{\mathbf{x}}$ is the filtered static state, and $\hat{\mathbf{y}} = [\hat{y}(1|1), \dots, \hat{y}(N|N)]$. The measurement error or *residual* term, ϵ , is expressed as:

$$\epsilon = \mathbf{y} - \hat{\mathbf{y}}, \quad (15)$$

or,

$$\epsilon = \mathbf{y} - \mathbf{H}\hat{\mathbf{x}}. \quad (16)$$

The *least-squares* estimate of \mathbf{x} is that value of $\hat{\mathbf{x}}(t|t)$ that minimizes J , where J is represented by the following equations:

$$J = \frac{1}{N} \sum_{k=1}^N [y_k - \hat{y}_k]^2. \quad (17)$$

In vector form, the quantity, J , to be minimized is expressed by :

$$J = (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}})^T (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}). \quad (18)$$

Setting $\partial J / \partial \hat{\mathbf{x}} = 0$, the following expression for the least-squares estimate, $\hat{\mathbf{x}}$, is obtained :

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y}. \quad (19)$$

The expression for the estimate, $\hat{\mathbf{x}}$, in equation (19) is obtained by using all of the available measurements y_1, \dots, y_N . This method of implementing the least-squares technique is referred to as *batch-processing*.

II.3.1.2 Recursive Least-Squares Method

The *batch-processing* method of implementing least-squares requires that the measurement data to be used at once. For on-line applications, it is more desirable to implement an estimation algorithm that uses the new measurements at a particular time instant to update the most recent estimates. This is done in the *recursive* least-squares method of estimation. To illustrate this, consider a scalar, static system defined by :

$$y(t) = x + v(t), \quad (20)$$

where, $y(t)$ is the measurement at time instant t , $v(t)$ is the noise term at time instant t and x is the state to be estimated. From equation (19), it can be shown that least-squares estimate, $\hat{x}(t|t)$, at time instant t is expressed as :

$$\hat{x}(t|t) = \frac{1}{t} \sum_{i=1}^t y(i). \quad (21)$$

When an additional measurement becomes available at time instant $(t+1)$, the new estimate becomes :

$$\hat{x}(t+1|t+1) = \frac{1}{t+1} \sum_{i=1}^{t+1} y(i). \quad (22)$$

Equation (22) can be rewritten to include the prior estimate, $\hat{x}(t|t)$, as follows :

$$\hat{x}(t+1|t+1) = \frac{t}{t+1} \left(\frac{1}{t} \sum_{i=1}^t y(i) \right) + \frac{1}{t+1} y(t+1), \quad (23)$$

or,

$$\hat{x}(t+1|t+1) = \frac{t}{t+1} \hat{x}(t|t) + \frac{1}{t+1} y(t+1) \quad (24)$$

Equation (24), can be rearranged further as follows :

$$\hat{x}(t+1|t+1) = \hat{x}(t|t) + \frac{1}{t+1} [y(t+1) - \hat{x}(t+1|t)], \quad (25)$$

where $\hat{x}(t+1|t) = \hat{x}(t|t)$. In other words, the best *prediction* of the constant state x up to and including the time instant t , $\hat{x}(t+1|t)$, is given by the estimate of the state x with the same measurement data, $\hat{x}(t|t)$. In equation (25), the term $y(t+1) - \hat{x}(t+1|t)$ is called the *residual* or the *innovations* term.

Therefore, the *recursive* least-square estimate of the state x at a time instant $(t+1)$, as given in equation (24), is solely given by the state estimate at the time instant t and the measurement at the time instant $(t+1)$. For large values of t , the following relation is satisfied (from equation (25)):

$$\hat{x}(t+1|t+1) - \hat{x}(t|t) \approx 0, \quad t \rightarrow \infty. \quad (26)$$

From equation (26), it is seen that the state estimate, $\hat{x}(t+1|t+1)$, converges to a steady-state value of the state estimate, \hat{x} . This is expressed as:

$$\hat{x}(t+1|t+1) \approx \hat{x}, \quad t \rightarrow \infty. \quad (27)$$

In this way, the state estimate \hat{x} for the linear static system is recursively obtained. The vector form of the recursive least-square estimate is similar to equation (24), and it can be similarly derived as in the scalar case [73].

II.3.2 Dynamic Systems

Nonadaptive methods of estimation, prediction and state filtering, for dynamic systems is discussed in this section. These methods assume that a system model is available. This model can be in either input-output or state-space form. However, state filtering methods exclusively use state-space system models.

II.3.2.1 Predictors

This section discusses prediction, either single-step-ahead prediction (SSP) or multi-step-ahead prediction (MSP), using dynamic models. Again, these models can be either in an input-output or state-space form.

II.3.2.1.1 Input-Output Models

The MIMO ARX model, represented by equation (8), is used to obtain the following SSP of the system output,

$$\begin{aligned}\hat{y}(t+1|t) = & A_1 y(t) + \cdots + A_{n_y} y(t - n_y + 1) \\ & + B_1 u(t) + \cdots + B_{n_u} u(t - n_u + 1),\end{aligned}\quad (28)$$

where $y(t)$ and $u(t)$ are the output and input (measurements), respectively, of the system. This form of equation (28), an SSP, is also called the *predictor* form of the MIMO ARX model.

Using the SISO ARX model, represented by equation (9), the following SSP of the system output, $\hat{y}(t+1|t)$, is obtained:

$$\begin{aligned}\hat{y}(t+1|t) = & a_1 y(t) + \cdots + a_{n_y} y(t - n_y + 1) \\ & + b_1 u(t) + \cdots + b_{n_u} u(t - n_u + 1),\end{aligned}\quad (29)$$

where $y(t)$ and $u(t)$ are the output and input (measurements), respectively, of the system. The SSP form of equation (29) is also called the *predictor* form of the SISO ARX model.

The MSP or *p-step-ahead-prediction* of the system output, $\hat{y}(t+1|t-p+1)$, given the measurement data until and including the time instant $(t-p+1)$ is obtained from the MIMO ARX model by rewriting it in the predictor form. If the MSP is performed recursively, then the form of the ARX model requires measurements at time instants $(t-p+2), \dots, t$. Assuming that $n_a < p$, which implies that the *predictions* of the MIMO ARX model must be used in place of the measurements at the time instants $(t-p+2), \dots, t$ because the measurements are available only until the time instant $(t-p+1)$. Thus, the multi-step-ahead MIMO ARX predictor is expressed as :

$$\begin{aligned} \hat{y}(t+1|t-p+1) = & \mathbf{A}_1 \hat{y}(t|t-p+1) + \dots + \mathbf{A}_{n_y} \hat{y}(t-n_y+1|t-p+1) \\ & + \mathbf{B}_1 \mathbf{u}(t) + \dots + \mathbf{B}_{n_u} \mathbf{u}(t-n_u+1). \end{aligned} \quad (30)$$

Similarly, the multi-step-ahead SISO ARX predictor is expressed as:

$$\begin{aligned} \hat{y}(t+1|t-p+1) = & a_1 \hat{y}(t|t-p+1) + \dots + a_{n_y} \hat{y}(t-n_y+1|t-p+1) \\ & + b_1 u(t) + \dots + b_{n_u} u(t-n_u+1). \end{aligned} \quad (31)$$

The MIMO single-step-ahead and multi-step-ahead ARX predictors are shown in Figure 9.

The SSP of the system output, $\hat{y}(t+1|t)$, given the measurement data until and including the time instant t is obtained from the MIMO ARMAX model by rewriting equation (10) in the *predictor* form :

$$\begin{aligned} \hat{y}(t+1|t) = & \mathbf{A}_1 \mathbf{y}(t) + \dots + \mathbf{A}_{n_y} \mathbf{y}(t-n_y+1) \\ & + \mathbf{B}_1 \mathbf{u}(t) + \dots + \mathbf{B}_{n_u} \mathbf{u}(t-n_u+1) \\ & + \mathbf{C}_1 \epsilon(t-1) + \dots + \mathbf{C}_{n_e} \epsilon(t-n_e+1), \end{aligned} \quad (32)$$

where $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are the outputs and inputs (measurements), respectively, of the process system, $\epsilon(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t|t-1)$ is the *prediction error* or *residual* term.

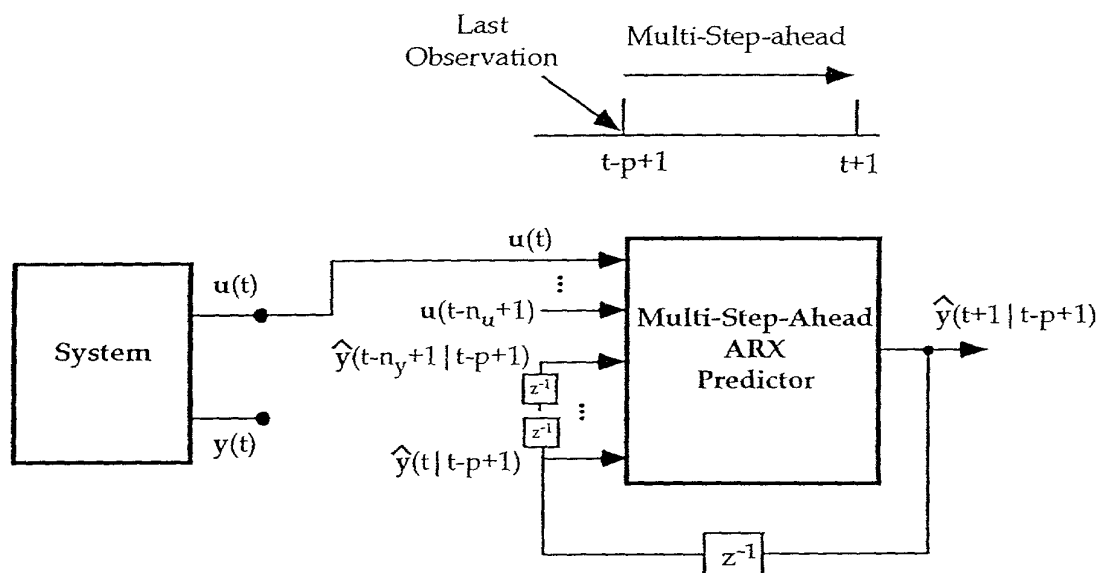
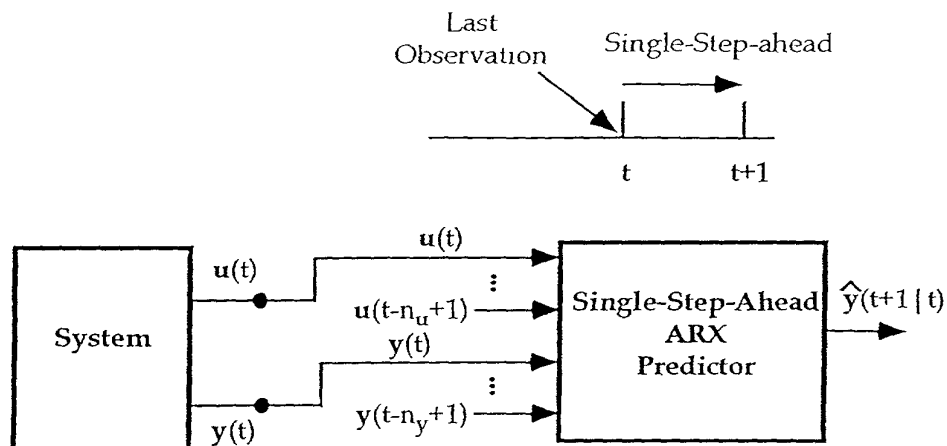


Figure 9. Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARX Predictors.

Similarly, the SSP of the system output, $\hat{y}(t+1|t)$, given the measurement data until and including the time instant t is obtained from the SISO ARMAX model by rewriting equation (11) in the *predictor* form :

$$\begin{aligned}\hat{y}(t+1|t) = & a_1 y(t) + \cdots + a_{n_y} y(t - n_y + 1) + b_1 u(t) + \cdots + b_{n_u} u(t - n_u + 1) \\ & + c_1 \epsilon(t) + \cdots + c_{n_e} \epsilon(t - n_e + 1).\end{aligned}\quad (33)$$

The multi-step-ahead ARMAX predictors, MIMO and SISO version, can be shown to be of the same form as the multi-step-ahead ARX predictors represented by equations (30) and (31) [43].

The MIMO single-step-ahead ARMAX predictor and the multi-step-ahead ARMAX predictor are shown in Figure 10.

Again, the reader should be reminded that the system inputs, $u(t)$, are assumed known throughout this research. If this is not the case, then estimates of the inputs, $\hat{u}(t)$, must be used instead.

II.3.2.1.2 State-Space Predictors

Assuming a *state-space* model form, the SSP of the system output $\hat{y}(t+1|t)$ is expressed by the following “predictor-type” equations [43]:

$$\hat{x}(t+1|t) = A \hat{x}(t|t-1) + B u(t) + K [y(t) - \hat{y}(t|t-1)], \quad (34)$$

$$\hat{y}(t+1|t) = C \hat{x}(t+1|t), \quad (35)$$

where, K is a gain matrix to be determined. Equations (34) and (35) are also referred to as the *innovations* representation of the state-space form. This latter term is the result of the innovation-type term present in equation (34).

II.3.2.2 State Filters

In its strict sense, state filtering became feasible by the introduction of the state-space representation and the resulting Luenberger and KF. The KF is a means of recursively obtaining an estimate, $\hat{x}(t+1|t+1)$, of the state $x(t+1)$ from the stochastic measurement data of a system, assuming the system is well-represented

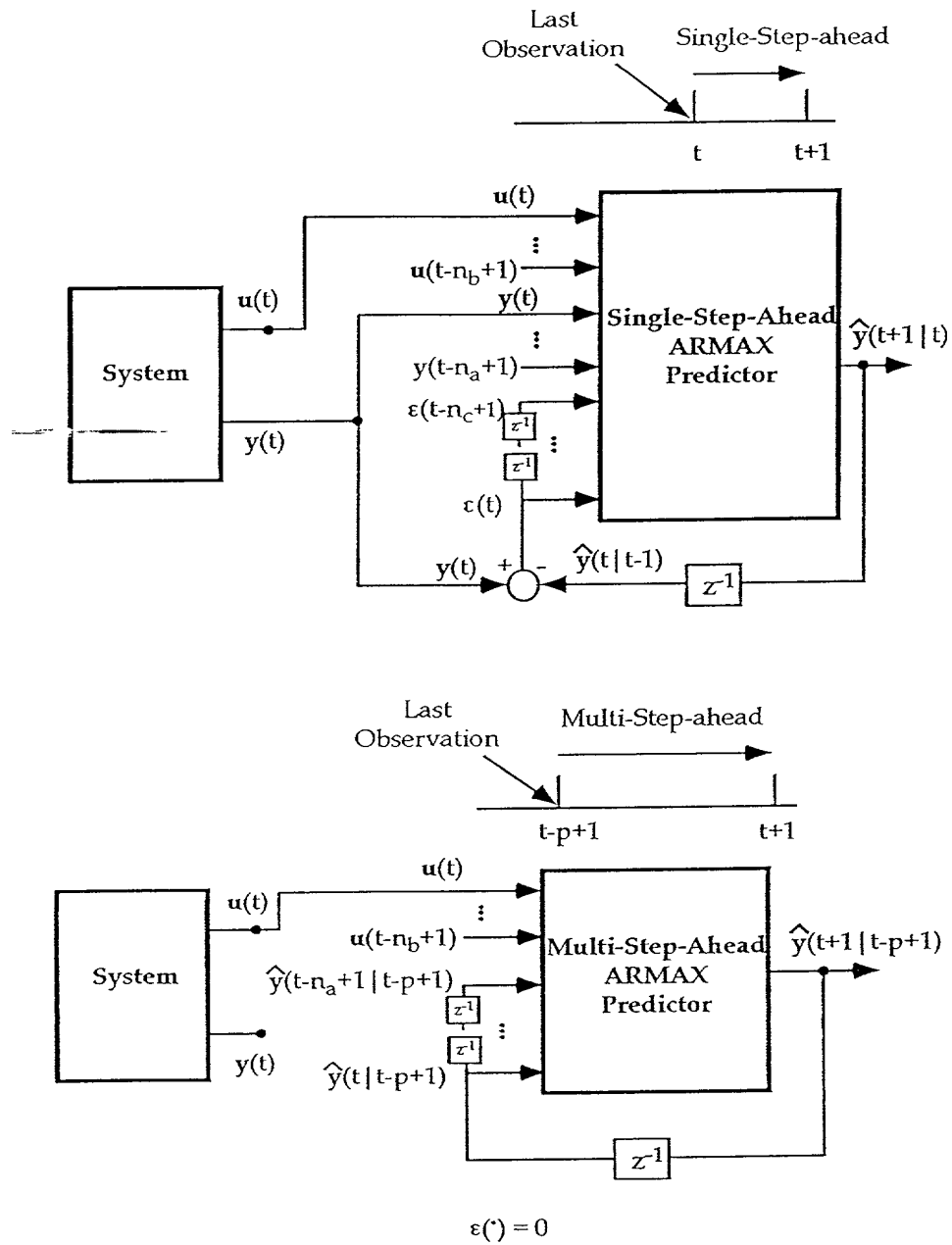


Figure 10. Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARMAX Predictors.

by a linear state-space model, as in equations (12) and (13). The KF equations are derived by assuming that the measurement and system noise, $w(t)$ and $v(t)$, are zero mean, white Gaussian processes with covariances \mathbf{Q} and \mathbf{R} , respectively, and zero cross-covariance [54].

To obtain the state estimate, $\hat{\mathbf{x}}(t+1|t+1)$, it is convenient to split the filter equations into *before* and *after* the measurement (sample) at the time instant $(t+1)$. It is assumed that the state estimate $\hat{\mathbf{x}}(t|t)$ is available from prior estimates or as initial conditions. Further the *error covariance matrix*, $\mathbf{P}(t|t)$, is defined as:

$$\mathbf{P}(t|t) = E \left\{ [\mathbf{x}(t) - \hat{\mathbf{x}}(t|t)] [\mathbf{x}(t) - \hat{\mathbf{x}}(t|t)]^T \right\}, \quad (36)$$

where $E\{\cdot\}$ is the *expectation* operator. $\mathbf{P}(t|t)$ is the solution of a steady-state Riccati equation [28], the details of which will not be discussed here. Nevertheless, the evolution of $\mathbf{P}(t|t)$ is described along with the rest of the filter equations. The KF is composed of a two-step procedure, involving a *prediction* and an *update* step.

Before the $(t+1)^{\text{th}}$ sample - Prediction Step :

The linear model equations, equations (12) and (13), are used to determine the predicted values of the state and output, $\hat{\mathbf{x}}(t+1|t)$ and $\hat{\mathbf{y}}(t+1|t)$, respectively, as follows [28]:

$$\hat{\mathbf{x}}(t+1|t) = \mathbf{A} \hat{\mathbf{x}}(t|t) + \mathbf{B} \mathbf{u}(t), \quad (37)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{C} \hat{\mathbf{x}}(t+1|t). \quad (38)$$

The predicted value of the error covariance matrix is given by:

$$\mathbf{P}(t+1|t) = \mathbf{A} \mathbf{P}(t|t) \mathbf{A} + \mathbf{Q}. \quad (39)$$

Equations (37) and (38) are referred to as the *predictor* equations.

After the $(t + 1)^{\text{th}}$ sample - Update Step :

Once the measurement $y(t + 1)$ is received, the predicted state value, $\hat{x}(t + 1|t)$, is corrected to account for “stochastic” errors in the state prediction part of the previous steps. The update equation for the error covariance matrix and the state are given as [28]:

$$P(t + 1|t + 1) = [C R^{-1} C + P^{-1}(t + 1|t)]^{-1}, \quad (40)$$

$$\hat{x}(t + 1|t + 1) = \hat{x}(t + 1|t) + K_{KF} [y(t + 1) - \hat{y}(t + 1|t)], \quad (41)$$

where,

$$K_{KF} = P(t + 1|t + 1) C R^{-1}, \quad (42)$$

and where K_{KF} is the KF gain matrix. The exact derivation of K_{KF} , although omitted in this discussion, is easily obtained from many good references [28],[32]. Equation (41) is referred to as the *state update* equation.

The evolution of the state estimate in the KF algorithm is shown in the timing diagram of Figure 11. A block diagram of the KF is shown in Figure 12.

Figure 12, depicting the KF, separates the *predictor* equations and *update* equations into two blocks. The SSP estimates (predicts) the state and output values at time instant $(t + 1)$ based on the measurement data available at the time instant t . The *state update* block then, at the time instant immediately after the measurement $(t + 1)$, corrects this state prediction based on the newly measured data, $y(t + 1)$, to estimate (filter) the state value at time instant $(t + 1)$. Therefore, the KF requires two steps, a state/output *prediction* step followed by an *update* step, in order to estimate (filter) the state values.

The KF algorithm has been the mainstay of state filtering methods since its inception over 30 years ago. In fact, the majority of state filtering literature in the last 25 years has dealt with more accurate and more efficient implementations of the original KF algorithm [28],[46].

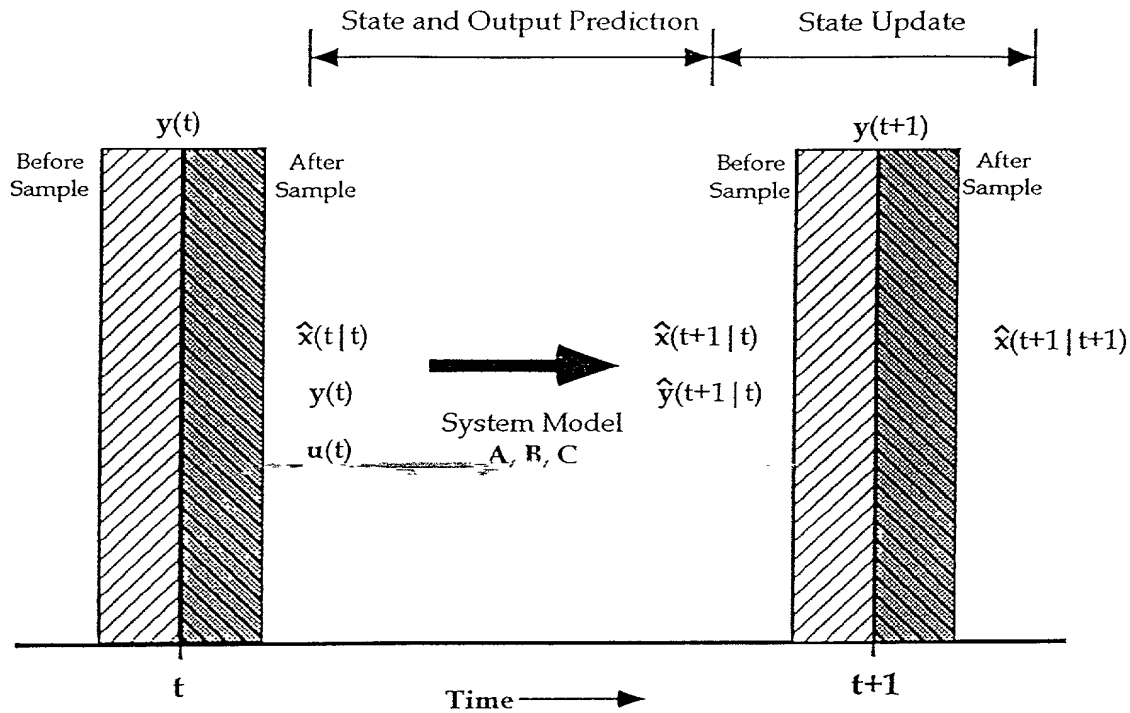


Figure 11. Kalman Filter Timing Diagram.

II.4 System Identification

Linear system identification (SI) refers to the procedure when the parameters of a linear system model must be determined from the measurement data [43],[69],[71]. This process is depicted in Figure 13. Linear SI methods can also be divided into *off-line* (batch and recursive) methods and *on-line* (recursive) methods.

In the following subsections, SI methods for SISO (ARX and ARMAX) models and state-space models are briefly described. SI methods for MIMO systems consists of repetitive applications of SI methods for SISO models.

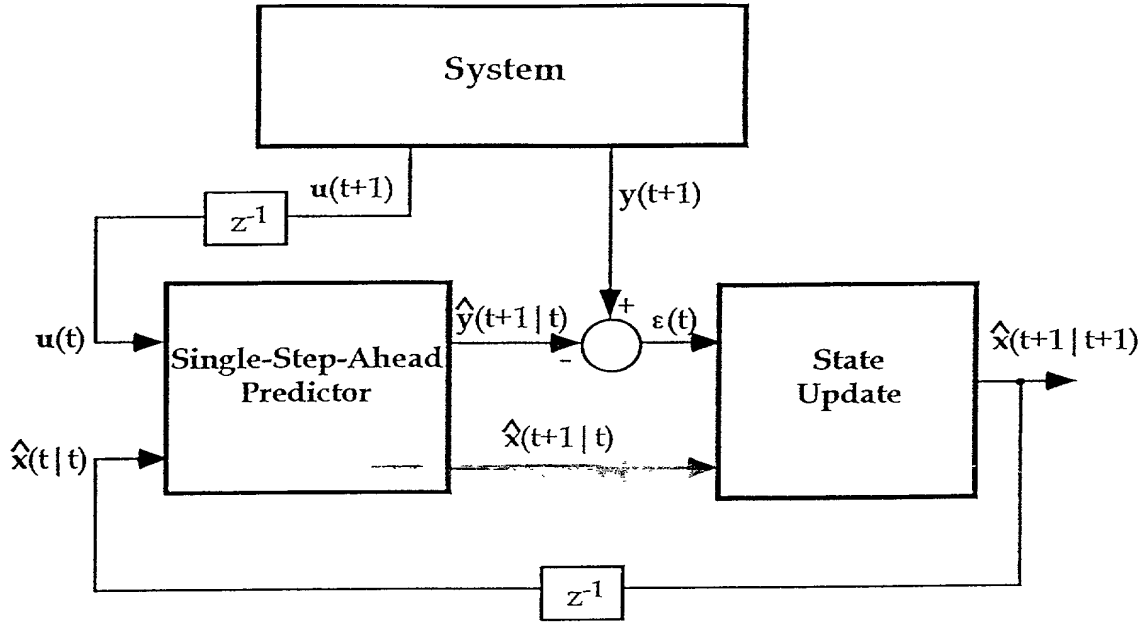


Figure 12. Schematic Diagram of the Kalman Filter.

II.4.1 Input-Output Models

The predictor form of an ARX model is represented by equation (29). It is now assumed that the parameters of the ARX model, $a_1, \dots, a_{n_y}, b_1, \dots, b_{n_u}$ are unknown and must be determined from the measurement data, $y(t)$ and $u(t)$. The ARX predictor form can be rewritten as:

$$\hat{y}(t+1|t; \theta) = \varphi^T(t+1) \theta, \quad (43)$$

where,

$$\varphi(t+1) = [y(t), \dots, y(t-n_y+1), u(t), \dots, u(t-n_u+1)]^T, \quad (44)$$

$$\theta = [a_1, \dots, a_{n_y}, b_1, \dots, b_{n_u}]^T. \quad (45)$$

Therefore, the ARX predictor form can be written as a scalar product between the data vector $\varphi(t+1)$ and the parameter vector θ . This is in the form of a *linear*

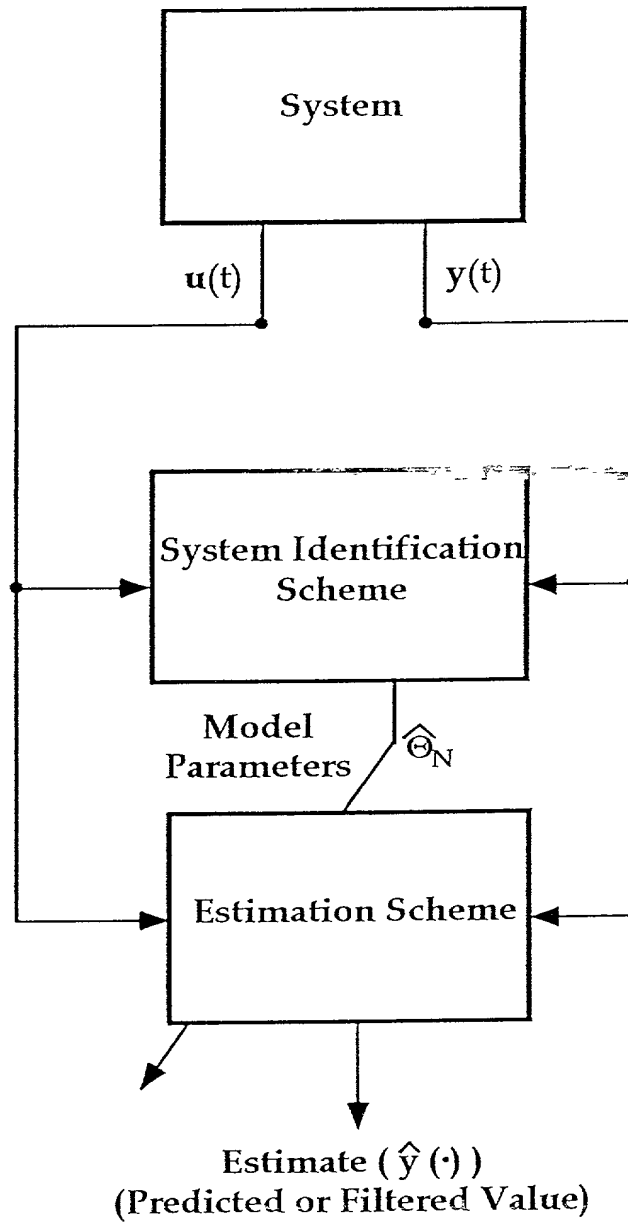


Figure 13. Block Diagram of Adaptive Filtering Using System Identification.

regression with the parameter vector θ as the *regression vector*. The least-squares method can be used to solve for θ in a non-iterative manner [43],[73].

The least-squares method of solving for the parameters of the ARX predictor is formulated by first defining the mean-square of the prediction error, $V_N(\theta, Z^N)$:

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^N [y(t) - \hat{y}(t|t-1; \theta)]^2, \quad (46)$$

where Z^N is the data set of N input-output samples $\{u(t), y(t)\}$ for $t = 1, \dots, N$. Equation (46) is also referred to as the *objective* function of the least-squares problem. The objective function, $V_N(\theta, Z^N)$, is minimized with respect to θ . The solution to the least-squares problem is the value of $\hat{\theta}_N$ that *minimizes* $V_N(\theta, Z^N)$, given by the following equation :

$$\hat{\theta}_N = \left[\sum_{t=1}^N \varphi(t) \varphi^T(t) \right]^{-1} \sum_{t=1}^N \varphi(t) y(t). \quad (47)$$

The predictor form of an ARMAX model is represented by equation (33). It is now assumed that the parameters of the ARMAX model, $a_1, \dots, a_{n_y}, b_1, \dots, b_{n_u}, c_1, \dots, c_{n_e}$ are unknown and have to be determined from the measurement data, $y(t)$ and $u(t)$. The ARMAX model predictor can be rewritten as :

$$\hat{y}(t+1; \theta) = \varphi^T(t+1; \theta) \theta, \quad (48)$$

where,

$$\varphi(t+1; \theta) = [y(t), \dots, y(t-n_y+1), u(t), \dots, u(t-n_u+1), \epsilon(t; \theta), \dots, \epsilon(t-n_e+1; \theta)]^T, \quad (49)$$

$$\theta = [a_1, \dots, a_{n_y}, b_1, \dots, b_{n_u}, c_1, \dots, c_{n_e}]^T, \quad (50)$$

and where,

$$\epsilon(t; \theta) = y(t) - \hat{y}(t|t-1; \theta). \quad (51)$$

Thus, the ARMAX predictor can be written as a scalar product between the data vector $\varphi(t+1; \theta)$ and the parameter vector θ . The data vector $\varphi(t+1; \theta)$ is now

dependent on the parameter vector θ due to the inclusion of the prediction error term, $\epsilon(t; \theta)$. Therefore, equation (48) is in the form of a *pseudo-linear regression* with the parameter vector θ as the *regression vector*. The least-squares method (batch or recursive) can be used to solve for θ as with the ARX predictor, however, no closed-form solutions are possible [43],[73].

II.4.2 State-Space Models

The state-space predictor is given by equations (34) and (35). The elements of the matrices A , B , C and K are assumed unknown and must be determined. Defining a parameter vector θ which contains all of the matrix elements of the state-space predictor, results in the following state-space predictor equations:

$$\hat{x}(t+1|t; \theta) = A(\theta) \hat{x}(t|t-1; \theta) + B(\theta) u(t) + K(\theta) [y(t) - \hat{y}(t|t-1; \theta)], \quad (52)$$

$$\hat{y}(t+1|t; \theta) = C(\theta) \hat{x}(t+1|t; \theta), \quad (53)$$

where

$$A(\theta) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1l} \\ a_{21} & a_{22} & \dots & a_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{ll} \end{bmatrix}, \quad B(\theta) = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{l1} & b_{l2} & \dots & b_{lm} \end{bmatrix}, \quad (54)$$

$$K(\theta) = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{l1} & k_{l2} & \dots & k_{ln} \end{bmatrix}, \quad C(\theta) = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1l} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nl} \end{bmatrix}, \quad (55)$$

and where the parameter vector θ comprises all the elements of the above matrices:

$$\theta = [a_{11}, \dots, a_{ll}, b_{11}, \dots, b_{lm}, k_{11}, \dots, k_{ln}, c_{11}, \dots, c_{nl}] \quad (56)$$

The state-space predictor, parametrized by the vector θ , as represented by equations (52) and (53) is referred to as the *directly parametrized innovations* form. Parameter estimation algorithms, such as least squares, can now be used to compute estimates of the parameter vector θ . Unfortunately, unless further simplifying assumptions are made regarding the structure of the matrices A , B , and C , such as

using one of the canonical forms, it is exceedingly difficult to obtain good parameter estimates, if any [43].

II.5 Adaptive Methods

In the preceding discussion regarding nonadaptive methods, it was assumed that the parameters of the linear models describing a system are known. If these parameters are not known, then they must be determined in some manner. Even if the parameters of a linear model had been determined at some prior time, the same linear model will no longer adequately describe the system because of changing system conditions, such as attrition of mechanical components, drifts etc.. Therefore, the parameters of the linear model must be recalculated. *Adaptation* refers to the process of determining or modifying (tuning) the parameters describing the system model using the measurement data of the system itself. Most system models require some form of adaptation over the life-cycle of the system, although the exact frequency of such adaptation is system dependent.

Therefore, in adaptive linear estimation the parameters of the involved models (either input-output or state-space) must be determined first using the SI methods discussed in the last section. After the system model parameters are determined, then linear estimation methods, also previously discussed can be applied. Linear adaptive state estimation methods, as with linear SI methods, can also be divided into *off-line* (batch and recursive) methods and *on-line* (recursive) methods. For more material on adaptive linear state estimation, the reader is referred to Goodwin and Sin [29] and Haykin [32].

II.6 Chapter Summary

In this chapter, an overview of linear estimation methods was presented. The commonly used static and dynamic system models were presented. The dynamic models presented were the most widely used input-output models, the ARX and the

ARMAX models, and state-space models. The linear estimation methods were categorized as nonadaptive prediction and filtering, system identification, and adaptive prediction and filtering methods. Among nonadaptive methods, where the system model is assumed to be known, SSP and MSP forms of the input-output models, the ARX and the ARMAX, was described. SSP forms of the state-space model was also described. The commonly used method of state filtering, the so-called KF, was also described. It was demonstrated that state filtering is effectively solved using a two-step procedure: the state/output prediction step, and a state update step. In SI, the methods of determining the parameters of the predictors based on the ARX, the ARMAX, and state-space models were described. The chapter concludes with adaptive methods of estimation, prediction and state filtering, in which the parameters of the system model are unknown or inaccurate and must be determined first.

NONLINEAR ESTIMATION METHODS-PREDICTION

III.1 Introduction

The majority of complex systems, in general, and process systems, in particular, are nonlinear in nature. In Chapter II, the discussed estimation methods, prediction and filtering, assume that the system model was *linear*. The advantage of using linear models in estimation, other than the ease of use, is that there is a wealth of knowledge dealing with linear estimation methods because of the maturity of linear analytic mathematical methods. The disadvantage of using linear models is that, in many real-world cases, these models are not accurate. For this reason, it is desirable to develop estimation methods using nonlinear system models. The advantage of using nonlinear models in estimation methods is that higher-fidelity nonlinear models will usually provide better estimates than linear models. However, this might not be always the case and it depends on the nature of the nonlinear model used and the system nonlinearities. The main disadvantage is that nonlinear models are often more difficult to identify and deal with than linear models, because of the paucity of systematic nonlinear identification techniques.

The main contribution of this research is in nonlinear state filtering, and the proposed method is described in Chapter IV. However, as discussed in Chapter II, state filtering is a two-step process, with the first step being a prediction step. Therefore, in order to better present the developments regarding nonlinear state filtering, first the methods and issues associated with nonlinear prediction are presented.

In this chapter, a general framework for nonlinear prediction methods is presented. This is followed by conventional function expansions, such as the polynomial Nonlinear Auto-Regressive with eXogeneous inputs (NARX) and polynomial Nonlinear Auto-Regressive Moving-Average with eXogeneous input (NARMAX), and also

NN function expansions of nonlinear predictors. Certain neural networks (NNs), both the Feedforward Multilayer Perceptron (FMLP) and the Recurrent Multilayer Perceptron (RMLP), are shown to be more general function expansions than the conventional polynomial expansions. Although the discussions of NNs are in the context of developing nonlinear predictors, more general inferences can be drawn regarding the applicability of NNs in approximating nonlinear functions. This motivates the rationale in using NNs in nonlinear state filtering problems, discussed in Chapter IV.

This chapter is organized as follows: section III.2 describes nonlinear input-output, and nonlinear state-space system models. Section III.3 describes a general framework for solving nonlinear prediction problems. Section III.4 describes predictors based on conventional function expansions, such as polynomial expansions (the polynomial NARX), used in nonlinear prediction. This section discusses non-adaptive methods, system identification, and adaptive methods using these models. Section III.5 describes NN expansions based on the FMLP and RMLP NNs. The general network equations for each of these two NNs are derived. Then, the FMLP and the RMLP is cast in the form of nonlinear predictors. This section discusses non-adaptive methods, NN learning algorithms, and adaptive methods using NNs models. Section III.6 summarizes the chapter.

III.2 System Models

Nonlinear estimation methods assume that the system under investigation is described by a nonlinear model. While this is a more accurate representation of a system (since most systems of interest are truly nonlinear in nature) than the one obtained by making the linearity assumptions described in the last section, some additional simplifying assumptions must be made on the nature of the nonlinear functions used. These assumptions are intended to transform the nonlinear model into a more mathematically tractable form, such that the traditional nonlinear estimation methods can be applied. This will be discussed in greater detail in the following subsections.

This section describes nonlinear system models, static and dynamic, commonly used in nonlinear estimation methods. As in the case of the linear models, two types of dynamic models are described, input-output models and state-space models.

III.2.1 Static Models

A nonlinear static system that relates the measurements $y(t)$ to the static state vector \mathbf{x} can be represented as follows:

$$\mathbf{y}(t) = \mathbf{H}(\mathbf{x}) + \mathbf{v}(t), \quad (57)$$

where $\mathbf{H}(\cdot)$ is a vector-valued nonlinear function and $\mathbf{v}(t)$ is the noise vector.

III.2.2 Dynamic Models

III.2.2.1 Input-Output Models

One of the most commonly used nonlinear input-output model is the Nonlinear Auto-Regressive with eXogenous inputs (NARX) model. The NARX model of a nonlinear system has the following representation:

$$\mathbf{y}(t+1) = \mathbf{f}(\mathcal{U}(t)) + \mathbf{e}(t+1), \quad (58)$$

where,

$$\mathcal{U}(t) \equiv [\mathbf{y}(t), \dots, \mathbf{y}(t - n_y + 1), \mathbf{u}(t), \dots, \mathbf{u}(t - n_u + 1)]^T, \quad (59)$$

and where,

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix}, \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}, \mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ \vdots \\ e_n(t) \end{bmatrix}, \quad (60)$$

are the system outputs, inputs, and noise respectively; n and m are number of outputs and inputs respectively; n_y and n_u are the maximum number of lags in the output and input, respectively; $\mathbf{e}(t)$ is the noise term, usually assumed to be a zero-mean, white Gaussian stochastic process; and $\mathbf{f}(\cdot)$ is a vector-valued nonlinear function (assumed known).

The NARX model, as represented by equation (58), can be written in terms of the individual inputs and outputs as follows [8]:

$$\begin{aligned} y_i(t+1) = & f_i\left(y_1(t), \dots, y_1(t-n_y^1+1), \dots, y_n(t), \dots, y_n(t-n_y^n+1), \right. \\ & \left. u_1(t), \dots, u_1(t-n_u^1+1), \dots, u_m(t), \dots, u_m(t-n_u^m+1)\right) \\ & + e_i(t+1), \quad \text{for } i = 1, \dots, n. \end{aligned} \quad (61)$$

A more general nonlinear input-output model is the Nonlinear Auto-regressive Moving Average with eXogenous inputs (NARMAX). The NARMAX model is represented by the following equation :

$$\mathbf{y}(t+1) = \mathbf{f}\left(\mathcal{U}_e(t)\right) + \mathbf{e}(t+1), \quad (62)$$

where,

$$\begin{aligned} \mathcal{U}_e(t) \equiv & [\mathbf{y}(t), \dots, \mathbf{y}(t-n_y+1), \mathbf{u}(t), \dots, \mathbf{u}(t-n_u+1) \\ & \mathbf{e}(t), \dots, \mathbf{e}(t-n_e+1)]^T, \end{aligned} \quad (63)$$

and where n_e is the maximum number of lags in the noise vector $\mathbf{e}(t)$.

III.2.2.2 State-Space Models

A general discrete-time nonlinear state-space model of a system can be written in the following form:

$$\mathbf{x}(t+1) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right) + \mathbf{w}(t), \quad (64)$$

$$\mathbf{y}(t) = \mathbf{h}\left(\mathbf{x}(t)\right) + \mathbf{v}(t), \quad (65)$$

where $\mathbf{y}(t)$ is the $n \times 1$ output vector of the nonlinear state-space model; $\mathbf{u}(t)$ is the $m \times 1$ input vector of the model; $\mathbf{x}(t)$ is the $l \times 1$ state vector of the nonlinear model; $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are vector valued nonlinear functions (assumed known); $\mathbf{w}(t)$, the process noise, is an $l \times 1$ vector; and $\mathbf{v}(t)$, the measurement noise, is an $n \times 1$ vector. As in the case of the linear system models, both $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are usually assumed to be zero-mean, white Gaussian stochastic vectors. A more general representation of a state-space model can be expressed as:

$$\mathbf{x}(t+1) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)\right), \quad (66)$$

$$\mathbf{y}(t) = \mathbf{h}\left(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t)\right) \quad (67)$$

Both state-space models, equations (64), (65), and equations (66), (67), represent the “noise” representation of state-space models.

III.3 Nonlinear Prediction: A General Framework

The general SSP form for the NARX model, represented by equation (61), can be written as follows:

$$\hat{y}_i(t+1|\bar{t}) = f_i\left(y_1(t), \dots, y_1(t-n_y^1+1), \dots, y_n(t), \dots, y_n(t-n_y^n+1), u_1(t), \dots, u_1(t-n_u^1+1), \dots, u_m(t), \dots, u_m(t-n_u^m+1)\right), \quad (68)$$

for $i = 1, \dots, n$. For an equal number of delayed inputs and outputs in each of the n outputs approximated, that is if $n_y^1 = n_y^2 = \dots = n_y^n = n_y$ and $n_u^1 = n_u^2 = \dots = n_u^m = n_u$, the following compact NARX SSP form is obtained:

$$\hat{y}_i(t+1|t) = f_i(\mathcal{U}(t)), \quad (69)$$

for $i = 1, \dots, n$, where $\mathcal{U}(t)$ was defined by equations (59). The MSP or *p-step-ahead prediction* of the output of the system output, $\hat{y}(t+1|t-p+1)$, given the measurement data until and including the time instant $(t-p+1)$ is obtained from the NARX model represented by equation (69). However, since the NARX predictor requires measurements at time instants $(t-p+1), \dots, t$, the *predictions* of the NARX model must be used instead. Therefore, the MSP form of the NARX model is expressed as:

$$\hat{y}_i(t+1|t-p+1) = f_i(\hat{\mathcal{U}}(t)), \quad \text{for } i = 1, \dots, n, \quad (70)$$

where

$$\hat{\mathcal{U}}(t) \equiv [\hat{y}(t|t-p+1), \dots, \hat{y}(t-n_y+1|t-p+1), \mathbf{u}(t), \dots, \mathbf{u}(t-n_u+1)]^T, \quad (71)$$

and where it is assumed that $n_y < p$. Block diagrams for the NARX SSP and MSP are shown in Figure 14.

Equations (69) and (70) represent a general form of the NARX-type nonlinear predictor equations, where the nonlinear functions $f_i(\cdot), i = 1, \dots, n$, must be determined. Similar arguments can be made and relevant expressions obtained for a general NARMAX-type nonlinear predictor.

III.4 Conventional Expansions

III.4.1 Polynomial Expansion

Most of the engineering literature in nonlinear estimation addresses, in some form or another, expansion of the nonlinearities involved in terms of some form of polynomial. In some instances, as it is the case with the polynomial NARX, higher order polynomial expansions are actually used in the prediction process. However, in most state filtering approaches, only the first (linear) term of the polynomial is utilized and re-computed repetitively. This issue will be revisited in Chapter IV. So most, if not all, conventional expansions approximate the nonlinearities present in equations (69) and (70) as follows:

$$f(x) \approx \sum_{i=0}^n \alpha_i x^i, \quad (72)$$

where x denotes the independent variable.

III.4.2 Nonadaptive Methods

III.4.2.1 Input-Output Models

Predictors based on nonlinear input-output models can be expressed in the form of a nonlinear regression. In these predictors, a regression vector is formed from linear and nonlinear combinations of the measurement data. The predictor itself, however, is *linear* in its parameters. Therefore, the parameters can be computed using some form of the recursive or batch least-squares algorithm.

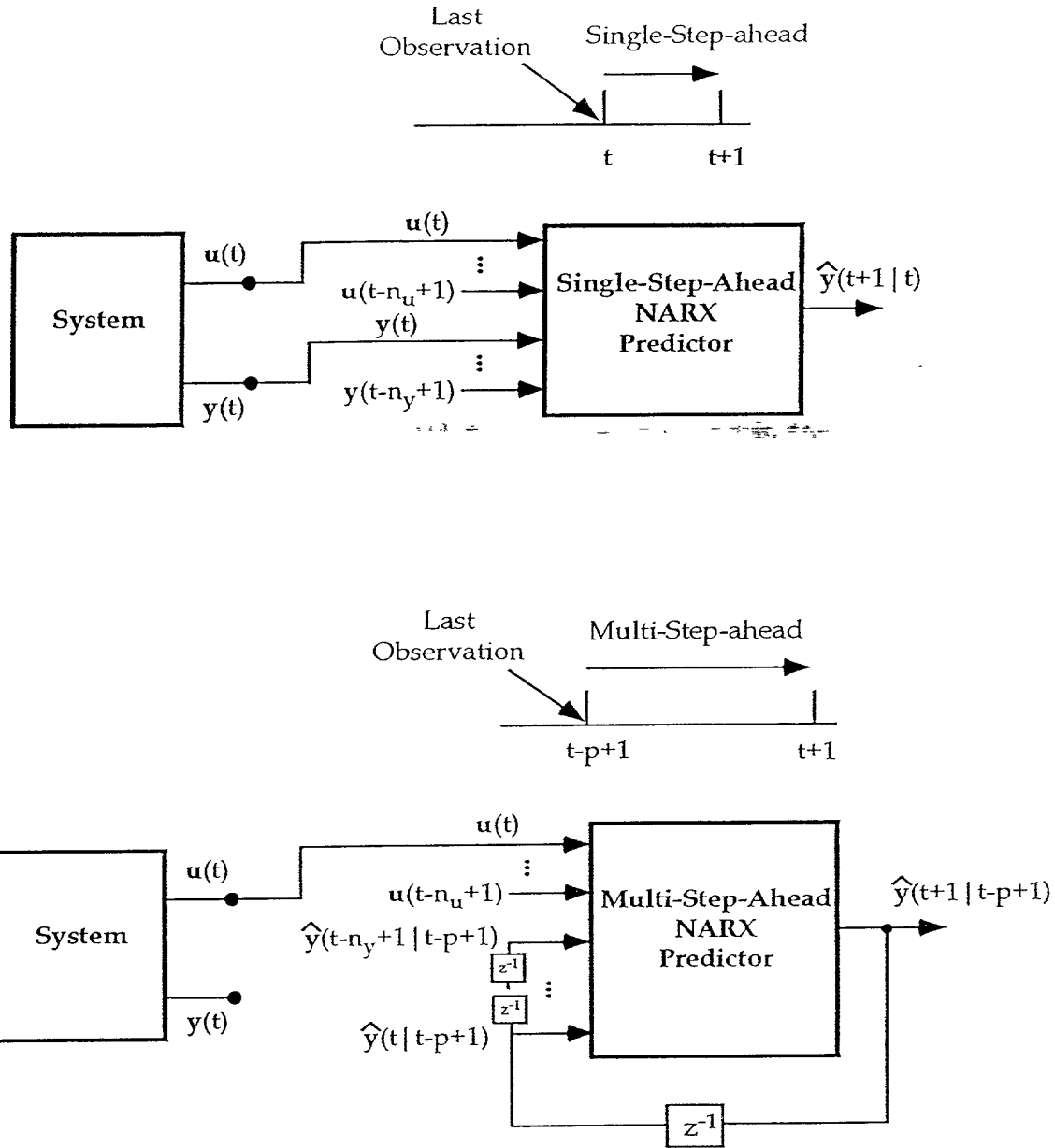


Figure 14. Block Diagram Representation of the Single-Step-Ahead and Multi-Step-Ahead NARX Predictors.

Many of the commonly used nonlinear predictors using input-output models are based on Volterra and Weiner series expansions [8], [43]. However these models are often too inflexible and suffer from the disadvantage of having too many parameters to be determined. The NARX and the NARMAX models are nonlinear input-output models that have been effectively used in a variety of nonlinear predictors without using a large number of parameters. Specifically, the *polynomial* NARX and NARMAX models are a special class of the general NARX and NARMAX models in which the involved nonlinear function is expanded using a polynomial basis. These models have been extensively studied by Billings et. al. [11],[12], and they are discussed in the following subsections.

III.4.2.1.1 NARX Models

The general representation of a NARX predictor is given by equation (69), where the nonlinear function $f_i(\cdot)$ is unknown and must be determined. Billings has proposed the following polynomial parametrization of the NARX predictor [11]:

$$\begin{aligned} \hat{y}_{\text{Poly},i}(t+1|t) = & \theta_0^{(i)} + \sum_{j_1=1}^q \sum_{j_2=j_1}^q \theta_{j_1 j_2}^{(i)} x_{j_1}(t) x_{j_2}(t) + \cdots \\ & + \sum_{j_1=1}^q \cdots \sum_{j_p=j_{p-1}}^q \theta_{j_1 \cdots j_p}^{(i)} x_{j_1}(t) \cdots x_{j_p}(t), \end{aligned} \quad (73)$$

for $i = 1, \dots, n$. In equation (73), p is the degree of the polynomial expansion, and where

$$q = n \times n_y + m \times n_u, \quad (74)$$

suggesting that an equal number of delayed input and output vector components is used in the expansion. Furthermore, the following expansion terms are defined:

$$\begin{aligned} x_1(t) = y_1(t), x_2(t) = y_1(t-1), \dots, x_{n \times n_y}(t) = y_n(t-n_y+1), \\ x_{n \times n_y+1}(t) = u_1(t), \dots, x_p(t) = u_m(t-n_u+1). \end{aligned} \quad (75)$$

The predictor structure represented by equation (73) is called the *polynomial NARX*.

III.4.2.2 State-Space Models

The SSP form of the nonlinear state-space model represented by equations (64) and (65) can be written as follows [71]:

$$\begin{aligned}\hat{\mathbf{x}}(t+1|t) &= \mathbf{f}\left(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t)\right) \\ &\quad + \mathcal{K}\left(\epsilon(t), \epsilon(t-1), \dots, \epsilon(t-n_e+1)\right),\end{aligned}\quad (76)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{h}\left(\hat{\mathbf{x}}(t+1|t)\right), \quad (77)$$

where $\hat{\mathbf{x}}(t+1|t)$ is the state SSP, $\hat{\mathbf{y}}(t+1|t)$ is the output SSP and $\mathcal{K}(\cdot)$, defined as the *filter gain function*, is a nonlinear vector valued function, $\epsilon(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t|t-1)$ is the *prediction error* or *innovations*, and n_e is the maximum number of lags of the prediction error.

III.4.3 Nonlinear System Identification

In this section, techniques for nonlinear SI applied to a nonlinear input-output model, the polynomial NARX, and nonlinear state-space models are discussed. It should be noted that nonlinear system identification methods, are also divided into *off-line* (batch and recursive) methods and *on-line* (recursive) methods.

III.4.3.1 Input-Output Models

The polynomial expansion of equation (73) can be rewritten in the following regression form [8]:

$$z_i(t+1) = \sum_{j=1}^M p_i(t+1)\theta_j, \text{ for } t = 1, \dots, N, \text{ and } i = 1, \dots, n, \quad (78)$$

where N is the data length, $z_i(t+1) = y_i(t+1)$, $p_j(t+1)$ are monomial $x_1(t)$ to $x_q(t)$ up to degree p , and θ_j are unknown parameters to be estimated, and M is the total number of terms in the polynomial. Therefore, each of the n system outputs can be independently approximated using the polynomial expansion of equation (73). Thus, the identification of an m -input and n -output system can be reduced to the identification of n , m -input single-output systems.

The orthogonal least-squares method is now used for parameter estimation because of the following reasons:

- (1) Parameter estimation methods using least-squares are well developed and applicable to the polynomial NARX model structure [13];
- (2) A nonlinear system can be expressed using a polynomial expansion, in which, the parameters enter linearly.

Equation (78) can now be written in the following matrix form:

$$\mathbf{z} = \mathbf{P} \boldsymbol{\theta}, \quad (79)$$

where,

$$\mathbf{z} = [z(1) \ z(2) \ \cdots \ z(N)]^T, \quad (80)$$

$$\mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2 \ \cdots \ \mathbf{p}_M], \quad (81)$$

$$\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_M]^T, \quad (82)$$

and where,

$$\mathbf{p}_i = [p_i(1) \ p_i(2) \ \cdots \ p_i(N)]^T, \quad \text{for } i = 1, \dots, M. \quad (83)$$

Thus the matrix \mathbf{P} is $N \times M$ dimensional. A linear least-squares estimation algorithm can now be applied to obtain the parameters $\hat{\boldsymbol{\theta}}$ which minimize $\|\mathbf{z} - \mathbf{P}\boldsymbol{\theta}\|$, where $\|\cdot\|$ is the Euclidian norm. It is well-known that the solution $\hat{\boldsymbol{\theta}}$ will satisfy the following equation [13]:

$$\mathbf{P}^T \mathbf{P} \hat{\boldsymbol{\theta}} = \mathbf{P}^T \mathbf{z}, \quad (84)$$

where $\mathbf{P}^T \mathbf{P}$ is called the information matrix. Further details regarding the orthogonal least-squares estimation method as applied to polynomial NARX can be found in the work of Billings, et. al. [8], [12], [13].

III.4.3.2 State-Space Models

Consider a nonlinear state-space model represented by equations (64) and (65). The SSP form of the nonlinear state-space model represented by the aforementioned equations can be parametrized as follows:

$$\begin{aligned}\hat{\mathbf{x}}(t+1|t; \theta) &= \mathbf{f}\left(\hat{\mathbf{x}}(t|t-1; \theta), \mathbf{u}(t)\right) \\ &\quad + \mathcal{K}\left(\epsilon(t; \theta), \epsilon(t-1; \theta), \dots, \epsilon(t-n_e+1; \theta)\right),\end{aligned}\quad (85)$$

$$\hat{\mathbf{y}}(t+1|t; \theta) = \mathbf{h}\left(\hat{\mathbf{x}}(t+1|t; \theta)\right), \quad (86)$$

where $\hat{\mathbf{x}}(t+1|t; \theta)$ is the state estimate (prediction), $\hat{\mathbf{y}}(t+1|t; \theta)$ is the output prediction, and $\mathcal{K}(\cdot)$, defined as the *filter gain function*, is a nonlinear vector valued function used in accounting for the innovations terms in the dynamics, $\epsilon(t; \theta) \equiv \mathbf{y}(t) - \hat{\mathbf{y}}(t|t-1; \theta)$ is the *prediction error* or *innovations*, and n_e is the maximum number of lags of the prediction error. Also, $\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$ and $\mathcal{K}(\cdot)$ are unknown nonlinear functions that are parametrized in terms of the (unknown) vector θ .

The majority of nonlinear state-space models are used in systems when there is a considerable amount of prior knowledge about the system model structure. This means that a first-principles (physical) model is available with some (usually only few) parameters to be determined. The approaches to these type of parameter identification problems are case dependent. There are no general solutions methods when empirical nonlinear state-space models are used. The best that can be done is to reformulate the nonlinear state-space model as an input-output model, such as the NARX or NARMAX models, and then apply the established nonlinear SI methods to these input-output models.

III.4.4 Adaptive Methods

In the preceding subsections, prediction techniques were discussed when the system model was nonlinear. In these techniques, it was assumed that the nonlinear model was known. However, if the nonlinear model is not known, then it must be somehow determined. Even if the nonlinear model is known initially, it may have to be

determined at a later point in time because of the changing conditions in the actual system. These changing conditions could be due to a variety of reasons including system drifts, deterioration of system components etc.. Adaptive prediction refers to the process of estimation when the nonlinear model must first be *identified* from the measurement data. Thus, it is the combined application of nonlinear SI and nonlinear prediction methods. There is very little that can be said regarding practical methods for dealing with nonlinear adaptive prediction methods using traditional expansions, such as polynomials. Nonlinear adaptive prediction methods, as with nonlinear SI methods, can be applied *off-line* (batch or recursively) and *on-line* (recursively).

III.5 Neural Network Expansions

NNs have been shown to be promising approaches in developing predictors for complex systems, especially when MSP performance becomes a model validation criterion [9],[14],[16], [49],[65]. The NNs considered in this study are the Feedforward Multilayer Perceptron (FMLP) and the Recurrent Multilayer Perceptron (RMLP). They are nonparametric and nonlinear model structures. Furthermore, the FMLP and RMLP represent a non-recurrent and a recurrent model structure, respectively. There are fairly systematic SI techniques that can be used to identify the parameters of the NN models. In NN terminology, the SI process is referred to as *training*, and the model parameters are referred to as the *weights* and *biases* of the NN model.

Now, it will be argued that NNs can be used in the context of nonparametric and nonlinear estimation and more specifically, in adaptive prediction, in this chapter, and in adaptive state filtering, in the following chapter. A brief overview of the NN properties in the context of nonlinear estimation is given in the following paragraphs.

III.5.1 Neural Network Descriptions

In this section, the FMLP and RMLP NNs are described. These two NN architectures are the building blocks upon which the prediction and state filtering methods are built. For a more detailed description than the one presented here, the reader is referred to [55],[56],[57].

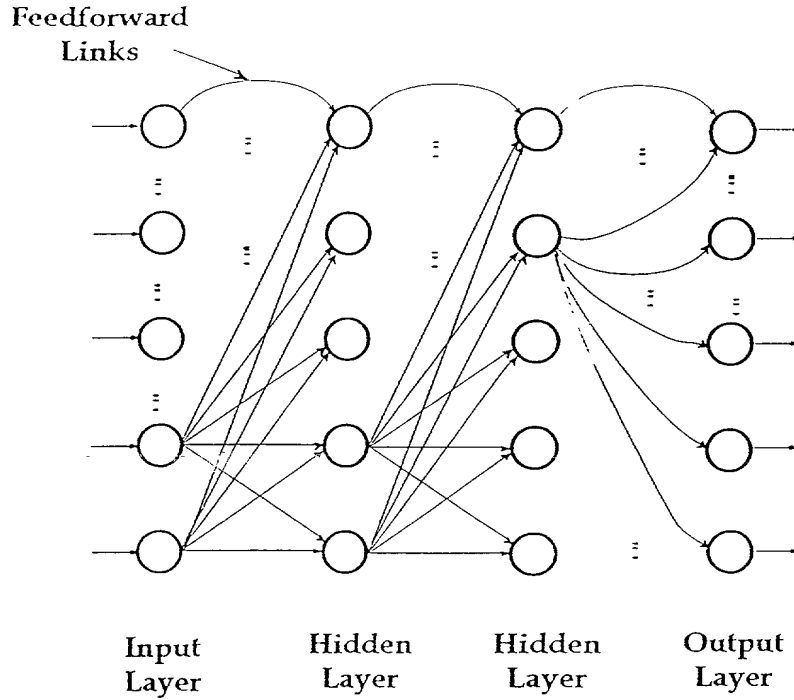


Figure 15. Schematic Diagram of the FMLP.

III.5.1.1 FMLP Networks

The NN shown in Figure 15 is called an FMLP. From an empirical modeling stand-point, it can be considered a nonlinear input-output model structure. That is, it is a static NN, and in the NN literature such networks are called feedforward or non-recurrent. The network is composed of an input layer, a series of hidden layers and an output layer. In this network, the signals from each node are transmitted to all the nodes in the next layer, and only the hidden layers have a sigmoid-type discriminatory function. The input and the output layers have linear discriminatory functions and the former has no biases. FMLPs with appropriate signals in the input layer are good at approximating static nonlinearities, i.e. memory-less nonlinear functions. Training of the network adjusts the active range of the discriminatory

function in the hidden layers, such that mostly the linear range is used for achieving a piecewise linear approximation [9].

Each of the processing elements of an FMLP network is governed by the following equation:

$$x_{[l,i]}(t) = \sigma_{[l,i]} \left(\sum_{j=1}^{N_{[l-1]}} w_{[l-1,j][l,i]} x_{[l-1,j]}(t) + b_{[l,i]} \right), \quad (87)$$

for $i = 1, \dots, N_{[l]}$ (the node index), and $l = 1, \dots, \mathcal{L}$ (the layer index), where $x_{[l,i]}(t)$ is the i th node output of the l th layer for sample t , $w_{[l-1,j][l,i]}$ is the weight (the adjustable parameter) connecting the j th node of the $(l-1)$ th layer to the i th node of the l th layer, $b_{[l,i]}$ is the bias (also an adjustable parameter) of the i th node in the l th layer, and $\sigma_{[l,i]}(\cdot)$ is the discriminatory function of the i -th node in the l -th layer. The unknown parameters of the FMLP, that is the weights and the biases, can be iteratively estimated using any one of the available training algorithms as it will be discussed in later sections [33], [30], [49].

III.5.1.2 RMLP Networks

The other NN architecture used in this study is the RMLP depicted in Figure 16.

The main distinction between the RMLP and the FMLP is the existence of recurrent and cross-talk (delayed) links in its hidden layers. These links feed the output of the node back to itself or to other nodes (cross-talk) with one time-delay. These time-delayed links thus impart memory to the node, and past information can be processed along with current node and, in general, network inputs. The output of each node of the RMLP is a function of its internal node state, its inputs, and indirectly the network inputs. In the NN literature the RMLP is referred to as the a recurrent network, because of its inherent feedback. Because of this inherent memory, resulting from the local feedback connections, an RMLP can be considered a nonlinear empirical state-space model structure, as demonstrated in the following sections. An empirical state-space model structure [43], as its name indicates, is a state-space representation in which the states themselves have no physical meaning or interpretation. In some

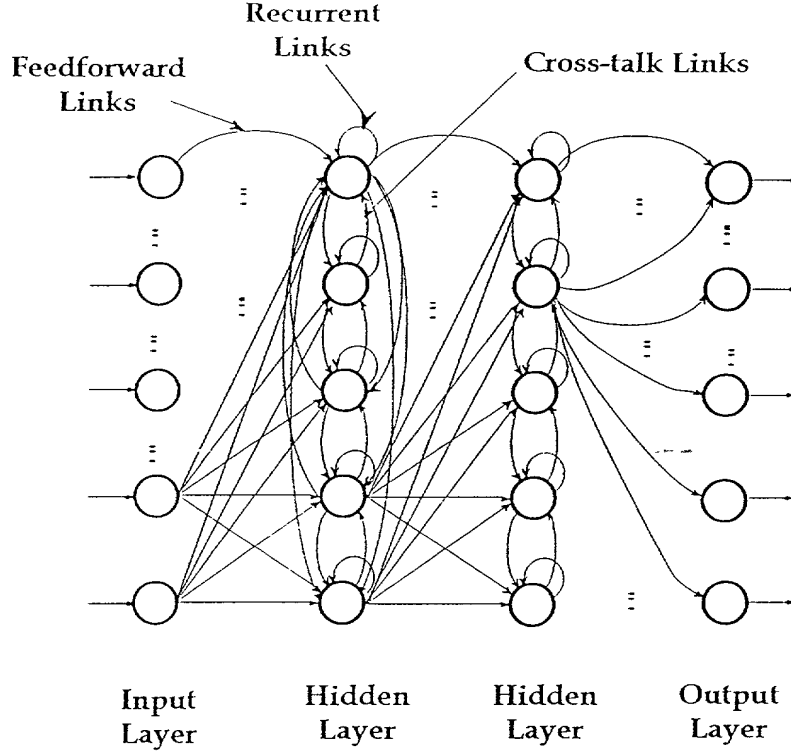


Figure 16. Schematic Diagram of the RMLP.

instances, an RMLP has been demonstrated to be more effective than an FMLP model structure in modeling complex nonlinear systems [16].

Each of the processing elements of the RMLP network is governed by the following difference equations:

$$z_{[l,i]}(t) = \sum_{j=1}^{N_{[l]}} w_{[l,j][l,i]} x_{[l,j]}(t-1) + \sum_{j=1}^{N_{[l-1]}} w_{[l-1,j][l,i]} x_{[l-1,j]}(t) + b_{[l,i]}, \quad (88)$$

and

$$x_{[l,i]}(t) = \sigma_{[l,i]}(z_{[l,i]}(t)), \quad (89)$$

where $z_{[l,i]}(t)$ represents the internal state variable of the i th node at the l th layer for sample t ; $x_{[l,i]}(t)$ is the i th node output of the l th layer for sample t , and $b_{[l,i]}$ is the

bias of the node ; $w_{[l,j][l',i]}$ is the weight associated with the link between the j th node of the l th layer to the i th node of the l' th layer. Furthermore, t represents the discrete-time at which the node and network outputs are computed, with the node index $i = 1, \dots, N_{[l]}$, and layer index $l = 1, \dots, \mathcal{L}$, and with the $\sigma_{[l,i]}(\cdot)$ for the input and output layers ($l = 1$ and $l = \mathcal{L}$) being linear. The term $b_{[l,i]}$ provides the bias for each node, defining the region where each node is “active”.

The RMLP was first proposed for use in nonlinear SI in 1990 [21],[56]. Since then, it has been extensively studied, computationally analyzed, and tested on numerous real-world applications. Furthermore, two learning algorithms for its effective training have been developed [16], [65]. An attractive feature of the RMLP is the explicit distinction among its layer, visually simplifying the separation of the feedforward and the feedback parts of the network. Although there is a substantial benefit gained by considering a (locally) recurrent network architecture for SI, as compared to using a purely feedforward one, there is, however, an increase in the complexity of the required learning. This is primarily attributed to the increasingly complex dynamic behavior of a recurrent network, the increased number of adjustable parameters for a given number of hidden nodes, and the issues associated with network stability during learning [16].

Another attractive feature of the RMLP, as well as any other recurrent neural network, is the fact that temporal information can be incorporated in it without the need to feed back any network outputs [16]. As a result, the RMLP can be useful for approximating nonlinear dynamic systems, i.e. nonlinearities with memory effects, as opposed to only static nonlinear systems. Even if global feedback (GF) is used in a NARX-type set-up, as discussed in the following sections, dynamic systems may be accurately approximated by purely feedforward networks, such as FMLPs. In some instances, numerous GF connections might be necessary to obtain good performance. With the appropriate choice of the number of layers and nodes, it is believed that an RMLP can approximate the dynamics of many complex system of interest in practical applications. Note, however, that there is no rigorous mathematical proof to justify

such a claim. Experience, though, with numerous complex systems has partially established the validity of this claim.

As with the FMLP, the inputs to the first layer of an RMLP, i.e. the network inputs, are denoted by $x_{[1]}$, and the output of the last layer, i.e. the network outputs, are denoted by $x_{[L]}$. The input layer of an RMLP network acts as a buffer for the input stream, whereas the hidden (discriminatory) layers enable the break-down of the function space to be identified into regions where it can be approximated by piecewise components. This is achieved by using a saturation function (usually a sigmoid or a hyperbolic tangent).

III.5.2 Nonadaptive Methods

Based on the developments of Section 3, a general NARX-type SSP can be expressed as follows:

$$\hat{y}(t+1|t) = f(\mathcal{U}(t)), \quad (90)$$

where, $\mathcal{U}(\cdot)$, $f(\cdot)$ and $\hat{y}(\cdot|\cdot)$ are as previously defined. Now, in view of the arguments made in section 3, and in order to utilize a NN as a predictor, the nonlinear function $f(\cdot)$ can be expanded in the following form:

$$f(\mathcal{U}(t)) \approx \sum_{i=1}^N w_i \sigma_i(\mathcal{U}(t)) + \mathbf{b}, \quad (91)$$

where, w_i is the i -th weight vector, $\sigma_i(\cdot)$ is the i -th scalar “squashing” basis function involved in the approximation, \mathbf{b} is the bias vector, and N is the number of terms used in the approximation. The number N represents, what, in the NN literature, is usually called hidden nodes. Furthermore, the nonlinear function $\sigma_i(\cdot)$ is any “squashing” function, such as a sigmoid, a hyperbolic tangent or a Gaussian. Defining, $\hat{y}_{\text{NN}}(t+1|t)$ as the prediction resulting from a NN approximation, and in view of equations (90) and (91), we can write:

$$\hat{y}_{\text{NN}}(t+1|t) = \sum_{i=1}^N w_i \sigma_i(\mathcal{U}(t)) + \mathbf{b}. \quad (92)$$

Figure 17 depicts the expansion of equation (92). Notice that the network depicted in Figure 17 has one input and one output layer, layers not containing any nonlinearities, and one hidden layer, layers containing the squashing nonlinear function. This expansion is simplified in two ways, as compared to the FMLP and RMLP architectures considered earlier. Specifically, only one hidden layer is considered, and the weights connecting the input layer to the hidden layer are set to one. These two simplifications result in a “linear-in-parameters” NN expansion, which expedites mathematical treatment.

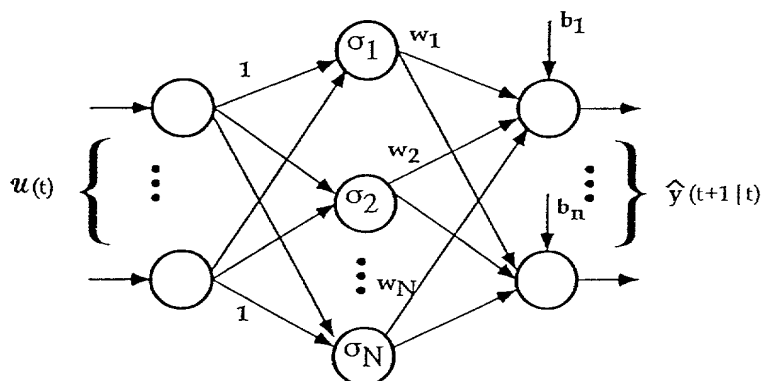


Figure 17. A Neural Network Predictor Representation.

More powerful predictors can be constructed by lifting the aforementioned two simplifying assumptions. The resulting predictors are mathematically complicated to express and analyze. Therefore, we now turn our attention to the “constructive” study and use of the two NN architectures, FMLP and RMLP as predictors.

III.5.2.1 FMLP Predictors

If the interest is in utilizing an FMLP as a nonlinear predictor, then it can be configured so that its input and output layer represent the measured and predicted system inputs and outputs, respectively. If the input layer of an FMLP is set to:

$$\mathbf{x}_{[1]}(t+1) \equiv \mathcal{U}(t), \quad (93)$$

then in view of equation (87), the equations for a single-hidden-layer FMLP network can be expressed as follows:

$$x_{[2,j]}(t+1) = \sigma_{[2,j]} \left(\sum_{i=1}^{N_{[1]}} w_{[1,i][2,j]} x_{[1,i]}(t+1) + b_{[2,j]} \right), \quad (94)$$

$$x_{[3,i]}(t+1) = \sum_{k=1}^{N_{[2]}} w_{[2,k][3,i]} x_{[2,k]}(t+1) + b_{[3,i]}, \quad (95)$$

$j = 1, \dots, N_{[2]}$, and $i = 1, \dots, N_{[3]}$, where for an n -output system, $N_{[3]} = n$, and where $N_{[2]}$ is the number of hidden layer nodes. Furthermore, we define

$$\hat{\mathbf{y}}_{\text{NN}}(t+1|t) \equiv \mathbf{x}_{[3]}(t+1), \quad (96)$$

where the vector $\mathbf{x}_{[3]}(t+1)$ is defined as:

$$\mathbf{x}_{[3]}(t+1) = \left[x_{[3,1]}(t+1), x_{[3,2]}(t+1), \dots, x_{[3,i]}(t+1), \dots, x_{[3,N_{[3]}]}(t+1) \right]^T. \quad (97)$$

Equation (94), (95) and (96) can now be combined in the following compact form:

$$\hat{\mathbf{y}}_{\text{NN}}(t+1|t) = \mathcal{F}(\mathcal{U}(t)), \quad (98)$$

where,

$$\mathcal{F}(\mathcal{U}(t)) = \mathcal{W}_{2 \rightarrow 3} \sigma_{[2]}(\mathcal{W}_{1 \rightarrow 2} \mathcal{U}(t) + \mathbf{b}_{[2]}) + \mathbf{b}_{[3]}, \quad (99)$$

where, $\mathcal{W}_{1 \rightarrow 2}$, $\mathcal{W}_{2 \rightarrow 3}$ are the weight matrices connecting the layers $1 \rightarrow 2$ and $2 \rightarrow 3$, respectively, and, $\mathbf{b}_{[2]}$, $\mathbf{b}_{[3]}$ are the bias vectors for layers 2 and 3, respectively, and where, $\sigma_{[2]}(\cdot)$ is the vector of discriminatory functions for layer 2. Equation (98) is

in the form of a SSP nonlinear (NARX) predictor, functionally similar to the form depicted by equation (90) [11],[16].

The same argument can be extended to a network with multiple hidden layers, and thus a general FMLP network can be configured in the form of a NARX SSP depicted by equation (69) [45]. In the NN literature, this approach of using the latest measurements in SSP is referred to as *Teacher Forcing* (TF).

Let us assume an FMLP with a total of \mathcal{L} layers, including an input and output layer with linear discriminatory functions, as depicted in Figure 15. Now, let us define the following function:

$$\mathcal{F}_{[i,j]}(\mathbf{x}_{[i]}(t+1)) = \sum_{k=1}^{N_{[i]}} w_{[i,k][i+1,j]} \sigma_{[i,k]}(x_{[i,k]}(t+1)) + b_{[i+1,j]}, \quad (100)$$

and,

$$\mathcal{F}_i = \left[\mathcal{F}_{[i,1]}, \mathcal{F}_{[i,2]}, \dots, \mathcal{F}_{[i,j]}, \dots, \mathcal{F}_{[i,N_{[i+1]}]} \right]^T, \quad \text{for } i = 1, \dots, (\mathcal{L} - 1), \quad (101)$$

where $\mathcal{F}_i(\cdot)$ represents the transformation of the input vector $\mathbf{x}_{[i]}(t+1)$ from layer i to layer $(i+1)$. The input vector $\mathbf{x}_{[i]}(t+1)$ is defined as:

$$\mathbf{x}_{[i]}(t+1) \equiv \left[x_{[i,1]}(t+1), \dots, x_{[i,j]}(t+1), \dots, x_{[i,N_{[i]}]}(t+1) \right]^T. \quad (102)$$

The rest of the notation is as described in earlier sections. Furthermore, the nonlinear (squashing) function is defined as follows:

$$\sigma_{[i,j]}(x) = \begin{cases} x, & \text{for } i = 1, \\ \tanh(x) & \text{for all other } i. \end{cases} \quad (103)$$

The graphical representation of the vector function $\mathcal{F}_i(\cdot)$ is depicted in Fig-

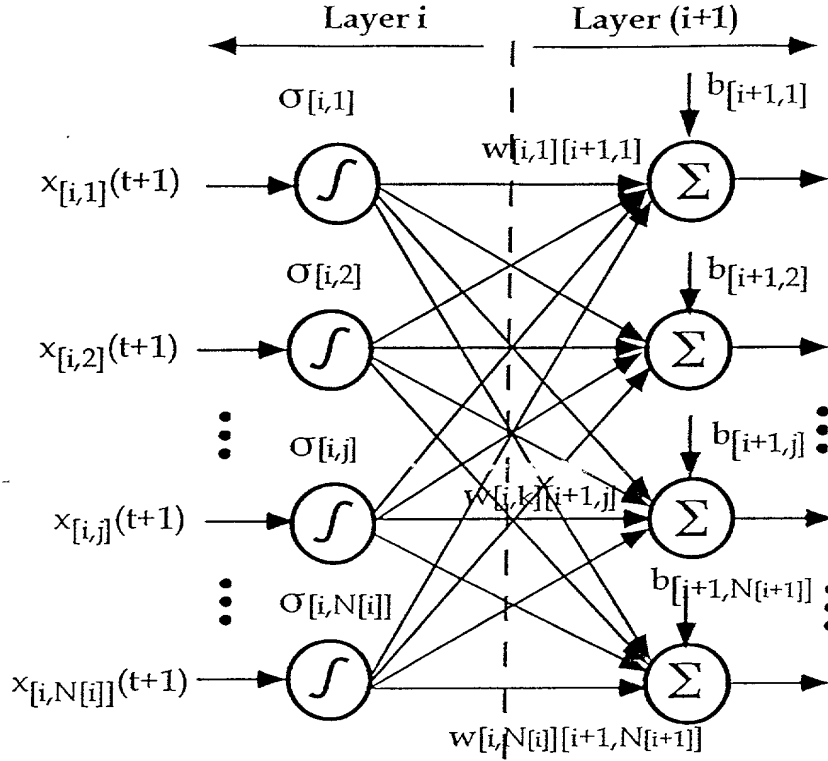


Figure 18. Graphical Representation of the FMLP Sublayers.

Figure 18. In view of equations (100) and (101) and considering that the output layer is linear, a general \mathcal{L} -layer FMLP can be written in the following compact SSP form:

$$\hat{y}_{NN}(t+1|t) = \mathcal{F}_{(\mathcal{L}-1)}\left(\mathcal{F}_{(\mathcal{L}-2)} \cdots \mathcal{F}_i\left(\mathcal{F}_{(i-1)} \cdots \mathcal{F}_1(\mathcal{U}(t))\right)\right), \quad (104)$$

where all variables are as previously defined.

It has been widely reported that the success of the FMLP networks in the input-output identification of nonlinear systems is partially because the approximation uses certain nonlinearities, e.g. hyperbolic tangent, sigmoid, etc., as basis functions. In fact, it is well-known that FMLPs are simply complex curve-fitting tools, allowing global approximation of most nonlinear functions that are of any practical use. Similar

convictions about the functionality of FMLP networks as good curve-fitting tools have been expressed by numerous other researchers [4], [5]. In fact, in a number of recent studies it has been rigorously proven that an FMLP network with one hidden layer is sufficient for approximating a large class of nonlinear functions. However, no information is provided regarding the number of regressors or nodes needed in the hidden layer [4], [5].

In the absence of high quality sensor readings, and in numerous other circumstances, the predictors of equation (104) can also be utilized in the following MSP form:

$$\hat{y}_{NN}(t+1|t-p+1) = \mathcal{F}(\hat{u}(t)), \quad (105)$$

and,

$$\hat{y}_{NN}(t+1|t) = \mathcal{F}_{(\mathcal{L}-1)}\left(\mathcal{F}_{(\mathcal{L}-2)} \cdots \mathcal{F}_i\left(\mathcal{F}_{(i-1)} \cdots \mathcal{F}_1(\hat{u}(t))\right)\right). \quad (106)$$

Equations (105) and (106) can be used for performing MSP, recursively, while equation (98) and (104) are primarily used for SSP. In the NN literature, the approach of using the past NN predictions as inputs to the NN itself is referred to as *Global Feedback* (GF). Block diagrams of the FMLP SSP and MSP are shown in Figure 19.

The free parameters of the FMLP model structure, that is its weights and its biases, can be iteratively estimated using what is known as learning algorithms [49]. However, in order to obtain accurate SSP and MSP, significantly different optimization problems must be formulated and solved for computing the weights and biases.

III.5.2.2 RMLP Predictors

If the RMLP is to be utilized as a nonlinear predictor, then its input and output layers must be appropriately defined. Because of the ever present uncertainties, predictors must be designed to operate in a closed-loop manner. This is the case with an RMLP-based predictor.

If the input layer of an RMLP is set to:

$$x_{[1]}(t+1) \equiv u(t), \quad (107)$$

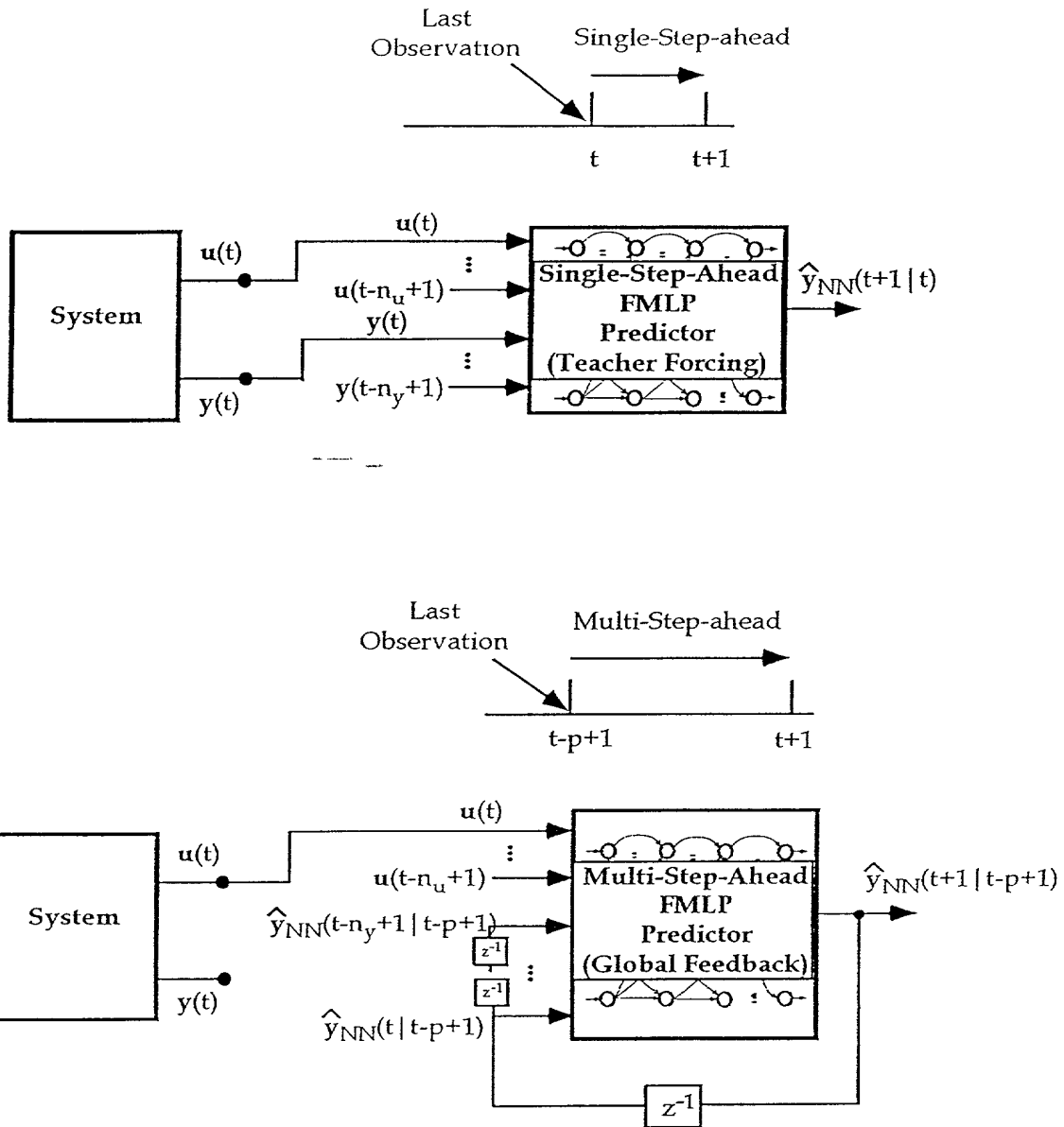


Figure 19. Block Diagrams of the FMLP Single-Step-Ahead and Multi-Step-Ahead Predictors.

then in view of equations (88) and (89), the equations for a single-hidden-layer RMLP network can be expressed as follows:

$$x_{[1,j]}(t+1) = z_{[1,j]}(t+1), \quad j = 1, \dots, N_{[1]}, \quad (108)$$

$$z_{[2,i]}(t+1) = \sum_{n=1}^{N_{[2]}} w_{[2,n][2,i]} x_{[2,n]}(t) + \sum_{n=1}^{N_{[1]}} w_{[1,n][2,i]} x_{[1,n]}(t+1) + b_{[2,i]}, \quad (109)$$

$$x_{[2,i]}(t+1) = \sigma_{[2,i]}(z_{[2,i]}(t+1)), \quad i = 1, \dots, N_{[2]}, \quad (110)$$

$$x_{[3,p]}(t+1) = \sum_{n=1}^{N_{[2]}} w_{[2,n][3,p]} x_{[2,n]}(t+1) + b_{[3,p]}, \quad p = 1, \dots, N_{[3]}, \quad (111)$$

where, $N_{[1]} = n \times n_y + m \times n_u$, and $N_{[3]} = n$. Furthermore, define

$$\hat{\mathbf{y}}_{\text{NN}}(t+1|t) \equiv \mathbf{x}_{[3]}(t+1), \quad (112)$$

where the vector $\mathbf{x}_{[3]}(t+1)$ is defined as in equation (97). Equations (108), (109), (110), (111), and (112) can be combined in the following compact form:

$$\hat{\mathbf{y}}_{\text{NN}}(t+1|t) = \mathcal{F}(\mathcal{U}(t), \mathbf{z}_{[2]}(t)), \quad (113)$$

where,

$$\begin{aligned} \mathcal{F}(\mathcal{U}(t), \mathbf{z}_{[2]}(t)) = & \mathcal{W}_{2 \rightarrow 3} \sigma_{[2]} \left(\mathcal{W}_{2 \rightarrow 2} \sigma_{[2]}(\mathbf{z}_{[2]}(t)) + \right. \\ & \left. \mathcal{W}_{1 \rightarrow 2} \mathcal{U}(t) + \mathbf{b}_{[2]} \right) + \mathbf{b}_{[3]}, \end{aligned} \quad (114)$$

where $\mathbf{z}_{[2]}(t)$ is the internal state vector of the hidden layer, and where all the other variables are as define in subsection 5.2.1.

Similar arguments can now be made regarding an RMLP with an arbitrary number of hidden layers. Again, assume an RMLP with a total of \mathcal{L} layers, as depicted in Figure 16, with linear, static input and output layers. Furthermore, let us define the following function:

$$\mathcal{F}_{[i,j]}(\mathbf{x}_{[i]}(t+1), \mathbf{z}_{[i+1]}(t)) \equiv \mathbf{z}_{[i+1,j]}(t+1), \quad (115)$$

where,

$$\begin{aligned}
 z_{[i+1,j]}(t+1) &= \sum_{k=1}^{N_{[i]}} w_{[i,k][i+1,j]} \sigma_{[i,k]}(z_{[i,k]}(t+1)) \\
 &\quad + (1 - \delta_{(i+1),\mathcal{L}}) \sum_{l=1}^{N_{[i+1]}} w_{[i+1,l][i+1,j]} \sigma_{[i,l]}(z_{[i+1,l]}(t)) \\
 &\quad + b_{[i+1,j]},
 \end{aligned} \tag{116}$$

and also define the following vector function resulting from definition (115):

$$\mathcal{F}_i = \left[\mathcal{F}_{[i,1]}, \mathcal{F}_{[i,2]}, \dots, \mathcal{F}_{[i,j]}, \dots, \mathcal{F}_{[i,N_{[i+1]}]} \right]^T, \quad \text{for } i = 1, \dots, (\mathcal{L} - 1), \tag{117}$$

where $\mathbf{x}_{[i]}(t+1)$ is the input vector of the transformation, as before, and $\mathbf{z}_{[i+1]}(t)$ is the output of the transformation $\mathcal{F}_{[i,j]}(\cdot)$ itself, at the previous time step. The delta function in equation (116) is defined as follows:

$$\delta_{i,j} = \begin{cases} 1, & i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{118}$$

All other variables are as defined for the FMLP predictor case. A graphical representation of this vector function is depicted in Figure 20. In Figure 20, the recurrent node inputs in layer i are not shown for clarity.

In view of equations (116) and (117), and the fact that the input and output layers are linear and static, a general \mathcal{L} -layer RMLP can be expressed in the following compact “input-output” SSP form, analogous to equation (113) for one hidden layer:

$$\begin{aligned}
 \hat{\mathbf{y}}_{\text{NN}}(t+1|t) &= \mathcal{F}_{(\mathcal{L}-1)} \left(\dots \mathcal{F}_i \dots \mathcal{F}_3 \left(\mathcal{F}_2 \left(\mathcal{F}_1(\mathcal{U}(t), \mathbf{z}_{[1]}(t)), \mathbf{z}_{[2]}(t) \right), \mathbf{z}_{[3]}(t) \right), \right. \\
 &\quad \left. \dots \mathbf{z}_{[i]}(t) \dots, \mathbf{z}_{[\mathcal{L}-1]}(t) \right),
 \end{aligned} \tag{119}$$

where all variables are as previously defined, and where the explicit dependence on the RMLP internal states is apparent. Replacing $\mathcal{U}(t)$ by $\hat{\mathcal{U}}(t)$ in equation (119), results in the “input-output” MSP form of the RMLP as follows:

$$\begin{aligned}
 \hat{\mathbf{y}}_{\text{NN}}(t+1|t) &= \mathcal{F}_{(\mathcal{L}-1)} \left(\dots \mathcal{F}_i \dots \mathcal{F}_3 \left(\mathcal{F}_2 \left(\mathcal{F}_1(\hat{\mathcal{U}}(t), \mathbf{z}_{[1]}(t)), \mathbf{z}_{[2]}(t) \right), \mathbf{z}_{[3]}(t) \right), \right. \\
 &\quad \left. \dots \mathbf{z}_{[i]}(t) \dots, \mathbf{z}_{[\mathcal{L}-1]}(t) \right).
 \end{aligned} \tag{120}$$

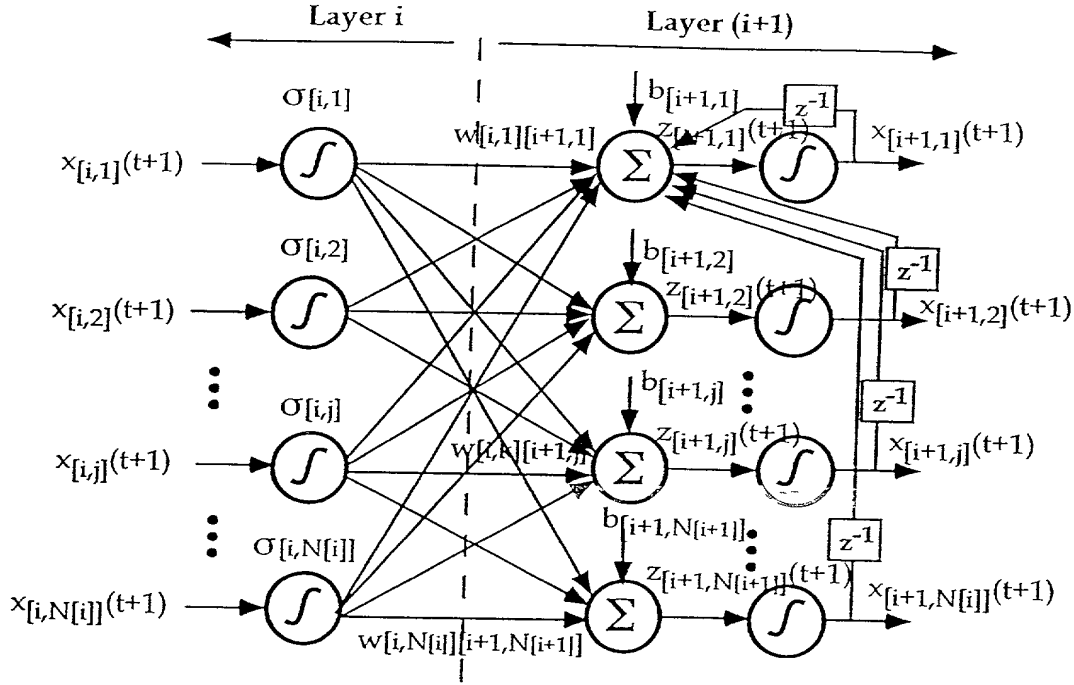


Figure 20. Graphical Representation of the RMLP Sublayers.

Equations (119) and (120) represent an “input-output” form of the RMLP SSP and MSP, though with the RMLP states present in the formulation.

Now, we turn our attention to a more transparent “empirical” state-space representation of the same predictor. This representation is made possible by the recurrent nature of the RMLP architecture. It is, however, expressed only for the case of a one hidden layer RMLP. By eliminating $x_{[1,j]}(t+1)$, $x_{[2,i]}(t+1)$, and $x_{[3,p]}(t+1)$ from equations (108) through (111), results in the following forms:

$$z_{[2,j]}(t+1) = \sum_{n=1}^{N_{[2]}} w_{[2,n][2,j]} \sigma_{[2,n]}(z_{[2,n]}(t)) + \sum_{n=1}^{N_{[1]}} w_{[1,n][2,j]} z_{[1,n]}(t+1) + b_{[2,j]}, \quad j = 1, \dots, N_{[2]}, \quad (121)$$

$$\begin{aligned}\widehat{y}_{\text{NN},p}(t+1|t) &= \sum_{n=1}^{N_{[2]}} w_{[2,n][3,p]} \sigma_{[2,n]}(z_{[2,n]}(t+1)) \\ &\quad + b_{[3,p]}, \quad p = 1, \dots, N_{[3]}.\end{aligned}\quad (122)$$

Now, considering that

$$\widehat{\mathbf{y}}_{\text{NN}}(t+1|t) = \left[\widehat{y}_{\text{NN},1}(t+1|t), \dots, \widehat{y}_{\text{NN},N_{[3]}}(t+1|t) \right]^T, \quad (123)$$

$$\mathbf{z}_{[1]}(t+1) = \mathbf{x}_{[1]}(t+1) \equiv \mathcal{U}(t), \quad (124)$$

$$\mathbf{z}_{[2]}(t+1) = [z_{[2,1]}(t+1), \dots, z_{[2,N_{[2]}]}(t+1)]^T, \quad (125)$$

and defining:

$$\mathbf{b}_{[2]} = [b_{[2,1]}, \dots, b_{[2,N_{[2]}]}]^T, \quad (126)$$

$$\mathbf{b}_{[3]} = [b_{[3,1]}, \dots, b_{[3,N_{[3]}]}]^T, \quad (127)$$

$$\mathcal{W}_{1 \rightarrow 2} = [w_{[1,i][2,j]}, \quad i = 1, \dots, N_{[1]}; j = 1, \dots, N_{[2]}], \quad (128)$$

$$\mathcal{W}_{2 \rightarrow 2} = [w_{[2,i][2,j]}, \quad i = 1, \dots, N_{[2]}; j = 1, \dots, N_{[2]}], \quad (129)$$

$$\mathcal{W}_{2 \rightarrow 3} = [w_{[2,i][3,j]}, \quad i = 1, \dots, N_{[2]}; j = 1, \dots, N_{[3]}], \quad (130)$$

equations (121) and (122) can be written in the following compact form :

$$\mathbf{z}_{[2]}(t+1) = \mathcal{W}_{2 \rightarrow 2} \sigma_{[2]}(\mathbf{z}_{[2]}(t)) + \mathcal{W}_{1 \rightarrow 2} \mathcal{U}(t) + \mathbf{b}_{[2]}, \quad (131)$$

$$\widehat{\mathbf{y}}_{\text{NN}}(t+1|t) = \mathcal{W}_{2 \rightarrow 3} \sigma_{[2]}(\mathbf{z}_{[2]}(t+1)) + \mathbf{b}_{[3]}, \quad (132)$$

where $\sigma_{[2]}(\cdot)$ is the following vector function:

$$\sigma_{[2]}(\cdot) = \left[\sigma_{[2,1]}(\cdot), \dots, \sigma_{[2,N_{[2]}]}(\cdot) \right]^T. \quad (133)$$

Equations (131) and (132) are of the form:

$$\mathbf{z}_{[2]}(t+1) = \mathcal{F}_{\text{SS}}(\mathbf{z}_{[2]}(t)) + \mathcal{W}_{1 \rightarrow 2} \mathcal{U}(t), \quad (134)$$

$$\widehat{\mathbf{y}}_{\text{NN}}(t+1|t) = \mathcal{H}_{\text{SS}}(\mathbf{z}_{[2]}(t+1)), \quad (135)$$

which is in the form of a nonlinear “empirical” SSP, as reported in Ljung [43]. In equations (134) and (135), $\mathcal{F}_{\text{SS}}(\cdot)$ and $\mathcal{H}_{\text{SS}}(\cdot)$ are nonlinear functions. It is an empirical

state-space because the state vector $\mathbf{z}_{[2]}(t+1)$ has no particular physical interpretation.

By replacing $\mathcal{U}(t)$ with $\widehat{\mathcal{U}}(t)$ in equation (134) the aforementioned state-space predictor can be utilized in the following MSP form:

$$\begin{aligned}\mathbf{z}_{[2]}(t+1) &= \mathcal{F}_{\text{SS}}(\mathbf{z}_{[2]}(t)) + \mathcal{W}_{1 \rightarrow 2} \widehat{\mathcal{U}}(t), \\ \widehat{\mathbf{y}}_{\text{NN}}(t+1|t-p+1) &= \mathcal{H}_{\text{SS}}(\mathbf{z}_{[2]}(t+1)).\end{aligned}\quad (136)$$

Block diagrams for the RMLP SSP and MSP are shown in Figure 21. While the preceding discussion about RMLP predictors assumed that the RMLP network had only one hidden layer, a similar representation can be developed for an arbitrary number of hidden layers. However, the resulting “empirical” state-space predictor (similar to equation (134)) is no longer explicit in the state variable, $\mathbf{z}(t)$, i.e. the RHS of the predictor contains terms of the form $\mathbf{z}(t+1)$.

In general, and in view of the statements made thus far, NNs (both FMLPs and RMLPs) are nonlinear, nonparametric models. The free parameters \mathcal{W} , both the weights and biases, are still to be determined, as explained in the next subsections.

In summary, and in view of equations (98), and (105), the SSP and MSP FMLP predictor can be expressed as

$$\widehat{\mathbf{y}}(t+1|t; \mathcal{W}) = \mathcal{F}(\mathcal{U}(t); \mathcal{W}), \quad (137)$$

and

$$\widehat{\mathbf{y}}(t+1|t-p+1; \mathcal{W}) = \mathcal{F}(\widehat{\mathcal{U}}(t); \mathcal{W}), \quad (138)$$

respectively, where the explicit dependence on the free parameters \mathcal{W} is denoted.

In the case of an RMLP, the predictors take a more complex form. A formulation similar to the one for an FMLP predictor can be expressed by considering equations (119) and (120). The SSP and MSP “input-output” predictor representations for the RMLP are:

$$\widehat{\mathbf{y}}(t+1|t; \mathcal{W}) = \mathcal{F}(\mathcal{U}(t), \mathbf{z}_{[1]}(t), \dots, \mathbf{z}_{[\mathcal{L}-1]}(t); \mathcal{W}), \quad (139)$$

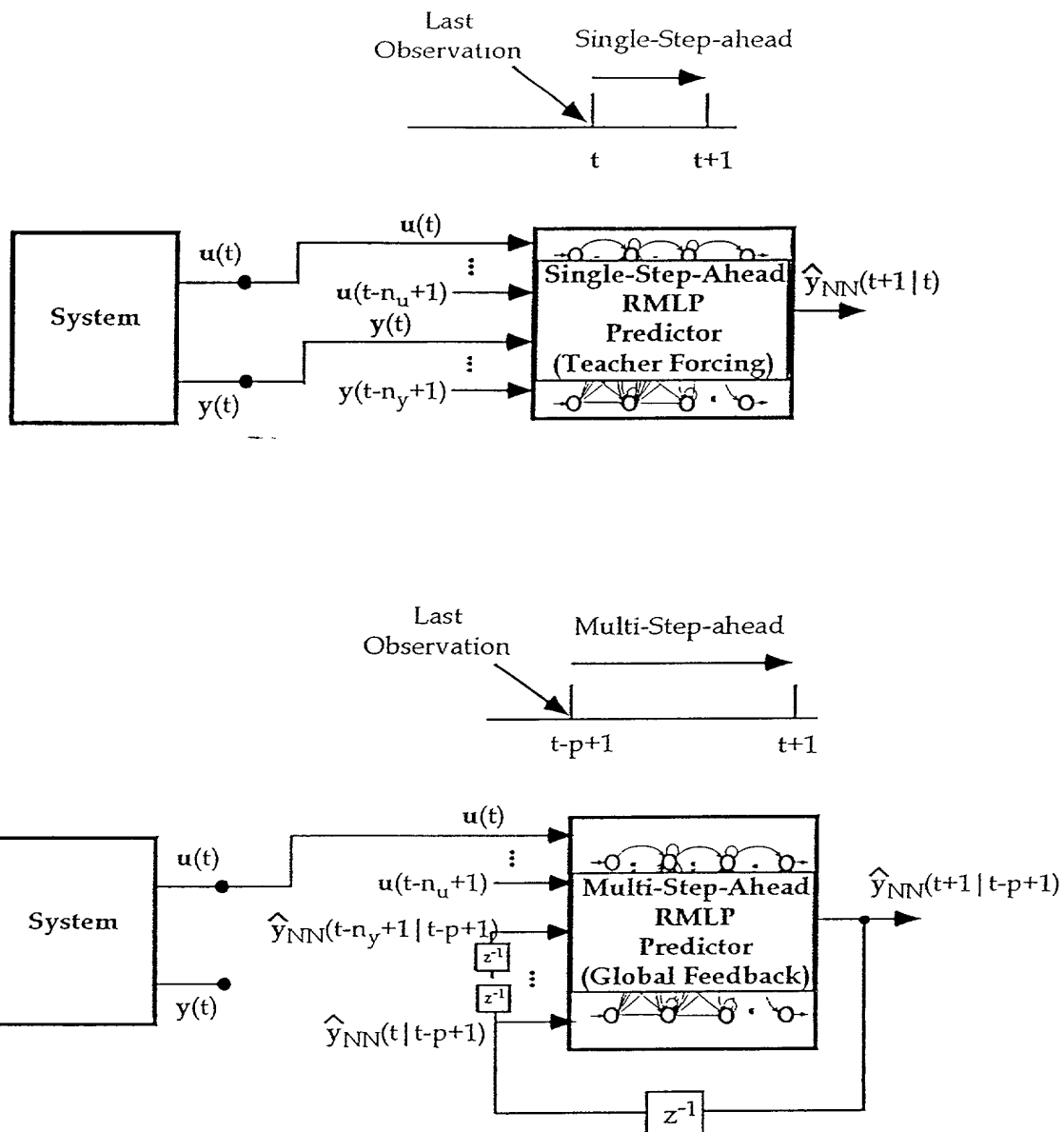


Figure 21. Block Diagrams of the RMLP Single-Step-Ahead and Multi-Step-Ahead Predictors.

and

$$\hat{\mathbf{y}}(t+1|t-p+1; \mathcal{W}) = \mathcal{F}(\hat{\mathbf{u}}(t), \mathbf{z}_{[1]}(t), \dots, \mathbf{z}_{[\mathcal{L}-1]}(t); \mathcal{W}), \quad (140)$$

where the dependence on the RMLP states is explicit. A more transparent representation of the RMLP SSP and MSP predictors, what is known as “empirical” state-space, can be presented by considering the single-hidden-layer RMLP equations (134) and (135).

Block diagrams for the SSP and MSP, using either an FMLP or RMLP NN, are shown in Figure 22.

III.5.3 System Identification : Neural Network Learning Algorithms

In both static and dynamic SI, and whether using NNs or not, the objective is to determine an algorithm, or rule, which adjusts the parameters of the model structure, based on the information contained in a given set of system observations. In the conventional literature this is called parameter estimation, whereas in the NN literature is called *learning*. Also, the term *target* is used to designate the system output observations used in training the NNs. When dealing with perceptron-type NNs, such as the FMLP, Backpropagation (BP) learning is the most commonly used learning method. It was first derived for static networks by Werbos [86], and later it was extended to incorporate temporal effects under the name of Backpropagation-Through-Time (BTT) (see Appendix A) [75]. Details on standard BP can be found in many books and papers which discuss NNs. There have been numerous variations to the standard BP algorithm reported in the literature, however, no attempt has been made in this study to document any of these approaches or to use them. For a brief review the reader is referred to Hecht-Neilsen [52] and Haykin [33].

However, learning algorithms as applied to RMLP-type NNs are not straightforward variations of the standard BP algorithm. This is because of the additional complexity imposed on the RMLP structure by the recurrency in the nodes. The next sections discuss learning algorithms as applied to the RMLP NNs with TF and GF.

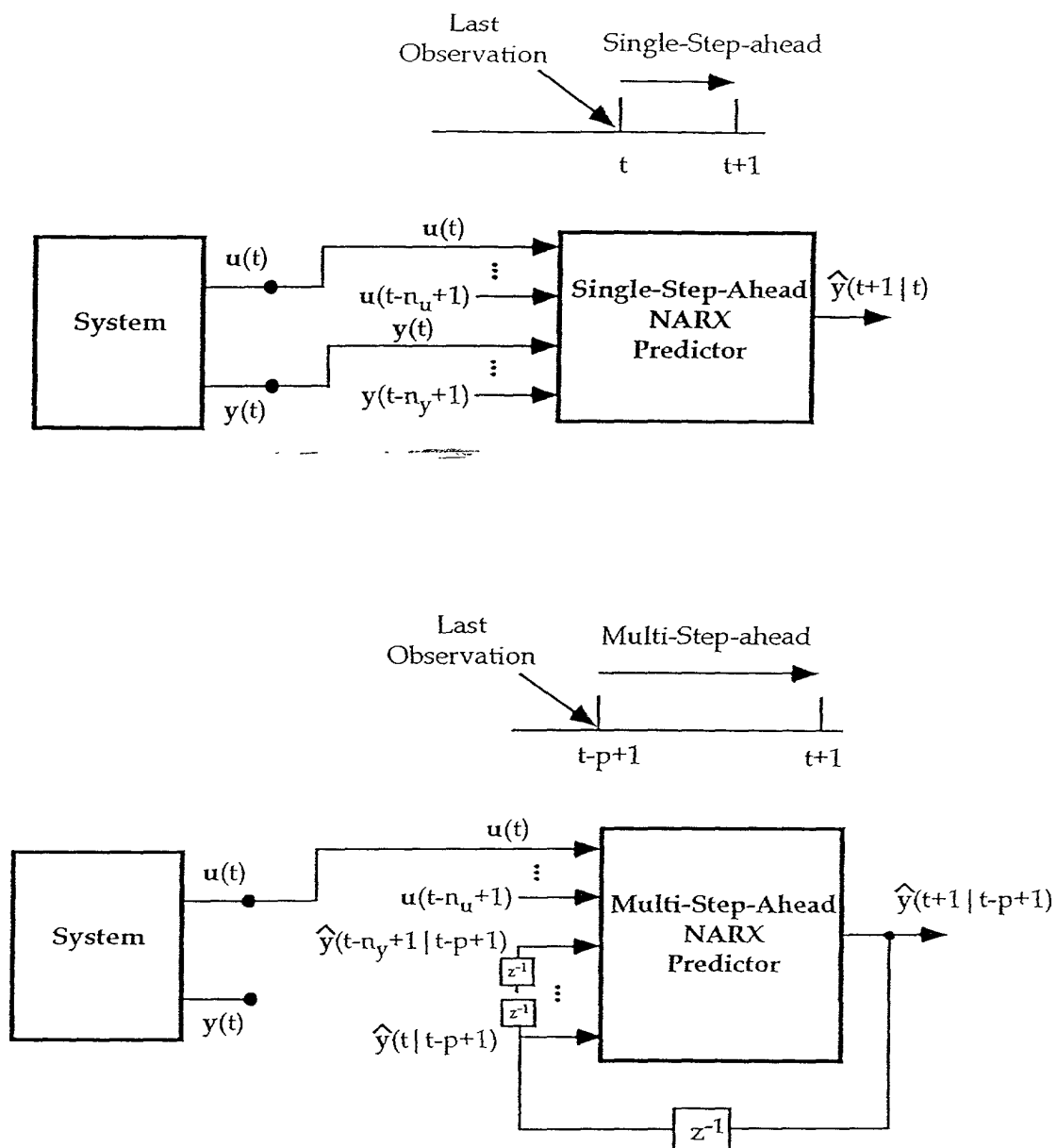


Figure 22. Block Diagrams of NN Single-Step-Ahead and Multi-Step-Ahead Predictors.

Furthermore, it should be noted that NN learning methods can also be divided into *off-line* (batch or recursive) methods and *on-line* (recursive) methods. While the cases presented in this study use off-line recursive methods, the algorithms presented can be easily adapted for on-line recursive operation.

III.5.3.1 RMLP Learning Algorithm with Teacher Forcing

The learning algorithm for RMLP with TF is a gradient descent algorithm minimizing a mean-squared-error (MSE) objective function. The basic mechanism of this learning rule is the adjustment of the network weights and the bias terms, until the MSE between the output predicted by the network and the sensed system output is less than a pre-specified tolerance.

Let us consider the off-line derivation of the algorithm for the TF case. The estimation data set can be expressed as follows:

$$S = \{(u_n(t), y_m(t)), \quad \forall t = 1, \dots, NP; n = 1, \dots, N_{[1]}; m = 1, \dots, N_{[L]}\}, \quad (141)$$

where NP is the total number of data pairs in the estimation data set. Specifically, the objective of most supervised learning algorithms is to determine the change in the network parameters $w_{[l-1,i][l,j]}$, $w_{[l,i][l,j]}$ and $b_{[l,i]}$, for all i, j and l , such that some form of the following function:

$$E \equiv \frac{1}{2} \sum_{t=0}^{NP-1} E(t+1) \equiv \frac{1}{2} \sum_{t=1}^{NP} \sum_{j=1}^{N_{[L]}} [\hat{y}_{NN,j}(t+1|t) - y_j(t+1)]^2, \quad (142)$$

is minimized.

Continuing under the assumption of off-line learning, the estimation data set is assumed to contain NP pairs of input-output data, which have been selected before initiating the training. These pairs are repetitively presented to the NN until it

reproduces them to within a desired error tolerance. During such training sessions, the network weights and biases are updated using the following gradient descent rules:

$$\Delta w_{[l-1,j][l,i]} = -\eta \sum_{t=0}^K \left(\frac{\partial E(t+1)}{\partial w_{[l-1,j][l,i]}} \right), \quad (143)$$

$$\Delta w_{[l,j][l,i]} = -\eta \sum_{t=0}^K \left(\frac{\partial E(t+1)}{\partial w_{[l,j][l,i]}} \right), \quad (144)$$

$$\Delta b_{[l,i]} = -\eta \sum_{t=0}^K \left(\frac{\partial E(t+1)}{\partial b_{[l,i]}} \right), \quad (145)$$

where K can be set to 0 or $(NP - 1)$ for individual or batch update, respectively, and where η is the learning rate, interactively set and altered by the user for proper convergence. For the on-line learning mode, however, there is no predetermined training set, and the weight updating must be performed as sensed information is received, similar to the conventional recursive SI.

For the dynamic gradient descent learning, the prediction error is defined by equation (142). The error gradient with respect to the weights and the biases can be obtained by using the chain-rule, as follows [16], [57]:

$$\frac{\partial E(t+1)}{\partial w_{[l,j][l,i]}} = 2 \sum_{k=1}^{N_{[C]}} \left[\hat{y}_{NN,k}(t+1|t) - y_k(t+1) \right] \frac{\partial \hat{y}_{NN,k}(t+1|t)}{\partial w_{[l,j][l,i]}}, \quad (146)$$

$$\frac{\partial E(t+1)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{k=1}^{N_{[C]}} \left[\hat{y}_{NN,k}(t+1|t) - y_k(t+1) \right] \frac{\partial \hat{y}_{NN,k}(t+1|t)}{\partial w_{[l-1,j][l,i]}}, \quad (147)$$

$$\frac{\partial E(t+1)}{\partial b_{[l,i]}} = 2 \sum_{k=1}^{N_{[C]}} \left[\hat{y}_{NN,k}(t+1|t) - y_k(t+1) \right] \frac{\partial \hat{y}_{NN,k}(t+1|t)}{\partial b_{[l,i]}}. \quad (148)$$

As an example for an update process of the feedforward weights, consider the gradient term $\frac{\partial \hat{y}_{NN,n}(t+1|t)}{\partial w_{[l,j][l,i]}}$. Differentiating equation (88) and equation (89) with respect to $w_{[l,j][l,i]}$, it can be seen that $\frac{\partial \hat{y}_{NN,n}(t+1|t)}{\partial w_{[l,j][l,i]}}$ can be obtained in terms of $\frac{\partial \hat{y}_{NN,m}(t|t-1)}{\partial w_{[l,j][l,i]}}$ and in terms of $\frac{\partial x_{[C-1,m]}(t+1)}{\partial w_{[l,j][l,i]}}$, for all m . Similarly, to obtain $\frac{\partial x_{[C-1,m]}(t+1)}{\partial w_{[l,j][l,i]}}$ first $\frac{\partial x_{[C-1,p]}(t)}{\partial w_{[l,j][l,i]}}$ and $\frac{\partial x_{[C-2,p]}(t+1)}{\partial w_{[l,j][l,i]}}$, for all p , must be evaluated. Therefore, the approach considered is to first evaluate $\frac{\partial x_{[2,m]}(t+1)}{\partial w_{[l,j][l,i]}}$, the output gradients of the second layer. Then, the

algorithm propagates forward, in the process evaluating the output gradients of the subsequent layers, until $\frac{\partial \hat{y}_{NN,n}(t+1|t)}{\partial w_{[l,j][l,i]}}$ is obtained. At that point the error gradients can be evaluated using equation (146) [16],[57].

The recursion equations used to execute the forward gradient propagation can be derived by differentiating equations (88) and (89) with respect to $w_{[l,j][l,i]}$, $w_{[l-1,j][l,i]}$ and $b_{[l,i]}$, respectively. These differentiations result in the following equations [16], [57]:

$$\frac{\partial x_{[l',n]}(t+1)}{\partial w_{[l,j][l,i]}} = \begin{cases} \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial w_{[l,j][l,i]}} \right] & \text{if } l' > l, \\ \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[l',j]}(t) \right] & \text{if } l' = l, \\ 0 & \text{if } l' < l, \end{cases} \quad (149)$$

$$\frac{\partial x_{[l',n]}(t+1)}{\partial w_{[l-1,j][l,i]}} = \begin{cases} \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } l' > l, \\ \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[l'-1,j]}(t+1) \right] & \text{if } l' = l, \\ 0 & \text{if } l' < l, \end{cases} \quad (150)$$

$$\frac{\partial x_{[l',n]}(t+1)}{\partial b_{[l,i]}} = \begin{cases} \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial b_{[l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial b_{[l,i]}} \right] & \text{if } l' > l, \\ \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial b_{[l,i]}} \right. \\ \left. + \delta_{in} \right] & \text{if } l' = l, \\ 0 & \text{if } l' < l, \end{cases} \quad (151)$$

for all layers $l', l = 1, \dots, \mathcal{L}$, and $i, j = 1, \dots, N_{[l]}$, sweeping the entire network, where δ_{in} is defined by equation (118), and where the initial values for the gradients $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l-1,j][l,i]}}$ and $\frac{\partial x_{[l',n]}(0)}{\partial b_{[l,i]}}$ are set to zero. Equations (149) through (151) must be applied

separately for each weight, and bias in the network. However, it is sufficient to always start from layer $l' = l$, and then propagate forwards until $l' = \mathcal{L}$, because $\frac{\partial x_{[l',n]}^{(t+1)}}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}^{(t+1)}}{\partial w_{[l-1,j][l,i]}}$ and $\frac{\partial x_{[l',n]}^{(t+1)}}{\partial b_{[l,i]}}$ equal 0 for $l' < l$ [16],[57].

III.5.3.2 RMLP Learning Algorithm with Global Feedback

The learning algorithm with GF is also a gradient descent algorithm minimizing a MSE objective function. GF means that the inputs utilized by the network include up to and including the latest predicted output(s) generated by the NN. Figure 21 shows a representation of the RMLP with GF, also depicted by equation (136), for a single hidden layer. In this figure $\hat{y}_{\text{NN}}(t+1|t-p+1)$ is the output predicted by the NN, while $\hat{y}_{\text{NN}}(t|t-p+1)$ is the output predicted by the NN at the previous time-step.

The fact that past predicted NN outputs utilized in GF, as compared to utilizing past sensed system output in TF, makes the gradients derivation in GF more complex than in TF. When using GF, the error gradients at each time-step, with respect to some of the weights and biases, depend on the values of the same gradients at the previous time-step. Thus, by recursively computing the error gradients, their dependence to all previous values of the error gradients is implicitly ensured. This feature enables the NN to perform accurate MSP on the training set, as well as any other set not seen by the network, depending on the procedure followed in the predictor design process. This is because the objective function being minimized consists of the MSP error of the training data set. Specifically, consider the following error function based on equation (142):

$$E \equiv \frac{1}{2} \sum_{t=0}^{NP-1} E(t+1) \equiv \frac{1}{2} \sum_{t=0}^{NP} \sum_{j=1}^{N_{[\mathcal{L}]}} [\hat{y}_{\text{NN},j}(t+1|t=0) - y_j(t+1)]^2, \quad (152)$$

where, $\hat{y}_{\text{NN},j}(t+1|t=0)$ is the recursively predicted NN output, for $t = 0, \dots, (NP - 1)$, and $y_j(t+1)$ is the j^{th} target (sensed output) in the training set. In view of equation (136) and the representation of the MSP in Figure 21, it is clear that every

predicted output $\hat{y}_{NN,j}(t+1|t=0)$ in the training set indirectly depends on all previous predictions $\hat{y}_{NN,j}(t|t=0), \hat{y}_{NN,j}(t-1|t=0), \dots, \hat{y}_{NN,j}(t-n_y+1|t=0)$.

In order to derive the error gradients for the RMLP with GF, without having to unfold the network as it is done in the BTT, it is necessary to introduce an additional assumption. If the set-up of the RMLP MSP shown in Figure 21 is applied to the MSP training set, then it is observed that a number of NN inputs might be correlated with each other. For example, $\hat{y}_{NN}(t|t=0)$ is dependent upon $\hat{y}_{NN}(t-1|t=0)$ and so on. However, $\hat{y}_{NN}(t-1|t=0)$ is correlated to $\hat{y}_{NN}(t-2|t=0)$ among others. This complication can be circumvented in two ways; either by assuming that $\hat{y}_{NN}(t|t=0), \dots, \hat{y}_{NN}(t-n_y+1|t=0)$ are outputs of other RMLPs, i.e. unfolding the network as done in the BTT, or by assuming that

$$\left. \begin{aligned} \hat{y}_{NN}(t-1|t=0) &= y(t-1), \\ &\vdots \\ \hat{y}_{NN}(t-n_y+1|t=0) &= y(t-n_y+1). \end{aligned} \right\} \quad (153)$$

That is all past predictions, with the exception of the latest prediction $\hat{y}_{NN}(t|t=0)$, are replaced by their equivalent targets from the training set. Thus, the RMLP is composed of two types of inputs nodes; those that consist of all independent inputs, i.e. $y(t-1), \dots, y(t-n_y+1), u(t), \dots, u(t-n_u+1)$, and the prediction node $\hat{y}_{NN}(t|t=0)$. Note that even with the aforementioned assumption, it is still the MSP error that is computed during training.

In the following development we make the assumption that equation (153) is valid. Then, the equations describing the i th node of the first hidden layer of the RMLP network using GF, for the t^{th} sample of the training set defined by equation (141), are given by the following difference equations:

$$\begin{aligned} z_{[2,i]}(t+1) &= \sum_{j=1}^{N_{[2]}} w_{[2,j][2,i]} x_{[2,j]}(t) + \sum_{j=1}^{N_{[1]}-N_{[L]}} w_{[1,j][2,i]} x_{[1,j]}(t+1) \\ &+ \sum_{j=N_{[1]}-N_{[L]}+1}^{N_{[1]}} w_{[1,j][2,i]} \hat{y}_j(t|t=0) + b_{[2,i]}, \end{aligned} \quad (154)$$

and,

$$x_{[2,i]}(t+1) = \sigma_{[2,i]}(z_{[2,i]}(t+1)). \quad (155)$$

The equations describing the i th node of the l th layer, other than the first hidden layer, of the RMLP network using GF are given by the following difference equation:

$$z_{[l,i]}(t+1) = \sum_{j=1}^{N_{[l]}} w_{[l,j][l,i]} x_{[l,j]}(t) + \sum_{j=1}^{N_{[l-1]}} w_{[l-1,j][l,i]} x_{[l-1,j]}(t+1) + b_{[l,i]}, \quad (156)$$

and

$$x_{[l,i]}(t+1) = \sigma_{[l,i]}(z_{[l,i]}(t+1)), \quad (157)$$

for $l = 2, \dots, N_{[\mathcal{L}]}$. Note that comparing equation (154) with equation (156), an additional third term is present in the former equation which accounts for the latest NN prediction utilized in the input layer.

For the dynamic gradient descent learning with GF, the prediction error is defined by equation (152). The error gradient with respect to the weights and the biases can be obtained by using the chain-rule, as follows [65]:

$$\frac{\partial E(t+1)}{\partial w_{[l,j][l,i]}} = 2 \sum_{n=1}^{N_{[\mathcal{L}]}} \left[\hat{y}_{\text{NN},n}(t+1|t=0) - y_n(t+1) \right] \frac{\partial \hat{y}_{\text{NN},n}(t+1|t=0)}{\partial w_{[l,j][l,i]}}, \quad (158)$$

$$\frac{\partial E(t+1)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{n=1}^{N_{[\mathcal{L}]}} \left[\hat{y}_{\text{NN},n}(t+1|t=0) - y_n(t+1) \right] \frac{\partial \hat{y}_{\text{NN},n}(t+1|t=0)}{\partial w_{[l-1,j][l,i]}}, \quad (159)$$

$$\frac{\partial E(t+1)}{\partial b_{[l,i]}} = 2 \sum_{n=1}^{N_{[\mathcal{L}]}} \left[\hat{y}_{\text{NN},n}(t+1|t=0) - y_n(t+1) \right] \frac{\partial \hat{y}_{\text{NN},n}(t+1|t=0)}{\partial b_{[l,i]}}. \quad (160)$$

Note that these three equations are similar to the ones given for the case of TF learning algorithm. The recursion equations used to execute the forward gradient propagation can be derived by differentiating equations (154) through (157) with respect to $w_{[l,j][l,i]}$, $w_{[l-1,j][l,i]}$, and $b_{[l,i]}$, respectively. These differentiations result in equations as described in the following development [3]. For the gradients with respect to the network connections within the layer, namely, the recurrent and cross-talk

weights, we have the following equation, valid for all hidden layers and output layer, namely, $1 < l' \leq \mathcal{L}$ [65]:

$$\frac{\partial x_{[l',n]}(t+1)}{\partial w_{[l,j][l,i]}} = \begin{cases} \sigma'_{[l',n]} \left(z_{[l',n]}(t+1) \right) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial w_{[l,j][l,i]}} \right] & \text{if } l' \neq l, \\ \sigma'_{[l',n]} \left(z_{[l',n]}(t+1) \right) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial w_{[l,j][l,i]}} + \delta_{in} x_{[l',j]}(t) \right] & \text{if } l' = l. \end{cases} \quad (161)$$

For interlayer connections, namely $\frac{\partial x_{[l',n]}(t+1)}{\partial w_{[l-1,j][l,i]}}$, we have the following gradient which is computed somewhat differently than the previous gradient. For the first hidden layer ($l' = 2$), the gradient is given by the following equations [65]:

$$\frac{\partial x_{[2,n]}(t+1)}{\partial w_{[l-1,j][l,i]}} = \begin{cases} \sigma'_{[2,n]} \left(z_{[2,n]}(t+1) \right) \left[\sum_{m=1}^{N_{[2]}} w_{[2,m][2,n]} \frac{\partial x_{[2,m]}(t)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } 1 \leq j \leq L, l \neq 2, \\ \sigma'_{[2,n]} \left(z_{[2,n]}(t+1) \right) \left[\sum_{m=1}^{N_{[2]}} w_{[2,m][2,n]} \frac{\partial x_{[2,m]}(t)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[1,j]}(t+1) \right] & \text{if } 1 \leq j \leq L, l = 2, \\ \sigma'_{[2,n]} \left(z_{[2,n]}(t+1) \right) \left[\sum_{m=1}^{N_{[2]}} w_{[2,m][2,n]} \frac{\partial x_{[2,m]}(t)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=N_{[1]}-N_{[L]}+1}^{N_{[1]}} w_{[1,m][2,n]} \frac{\partial x_{[1,m]}(t)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } L < j \leq N_{[1]}, \forall l, \end{cases} \quad (162)$$

where $L \equiv N_{[1]} - N_{[L]}$, and,

$$x_{[1,m]}(t) = \hat{y}_{NN,k}(t|t-1), \text{ if } N_{[1]} - N_{[L]} < m \leq N_{[L]} \text{ and } 1 \leq k \leq N_{[L]}. \quad (163)$$

For all hidden layers other than the first hidden layer, namely $2 < l' \leq \mathcal{L}$, the gradient with respect to interlayer connections is computed as follows [65]:

$$\frac{\partial x_{[l',n]}(t+1)}{\partial w_{[l-1,j][l,i]}} = \begin{cases} \sigma'_{[l',n]} \left(z_{[l',n]}(t+1) \right) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial w_{[l-1,j][l,i]}} \right] & \text{if } l' \neq l, \\ \sigma'_{[l',n]} \left(z_{[l',n]}(t+1) \right) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial w_{[l-1,j][l,i]}} \right. \\ \left. + \delta_{in} x_{[l'-1,j]}(t+1) \right] & \text{if } l' = l. \end{cases} \quad (164)$$

The gradients with respect to the biases are given by the following equation:

$$\frac{\partial x_{[l',n]}(t+1)}{\partial b_{[l,i]}} = \begin{cases} \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial b_{[l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial b_{[l,i]}} \right] & \text{if } l' > l, \\ \sigma'_{[l',n]}(z_{[l',n]}(t+1)) \left[\sum_{m=1}^{N_{[l']}} w_{[l',m][l',n]} \frac{\partial x_{[l',m]}(t)}{\partial b_{[l,i]}} \right. \\ \left. + \sum_{m=1}^{N_{[l'-1]}} w_{[l'-1,m][l',n]} \frac{\partial x_{[l'-1,m]}(t+1)}{\partial b_{[l,i]}} + \delta_{in} \right] & \text{if } l' = l. \end{cases} \quad (165)$$

Equations (161) through (165), are valid for all layers l' , $l = 2, \dots, \mathcal{L}$, and nodes $i, j = 1, \dots, N_{[l]}$, sweeping the entire network, and where,

$$\delta_{in} = \begin{cases} 1 & \text{if } i = n, \\ 0 & \text{otherwise.} \end{cases} \quad (166)$$

Note that $N_{[1]}$ includes the number of current and delayed inputs of the system, as well as the number of delayed output(s) and the latest prediction utilized as input to the network. Furthermore, the initial values for the gradients $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l,j][l,i]}}$, $\frac{\partial x_{[l',n]}(0)}{\partial w_{[l,j][l,i]}}$, and $\frac{\partial x_{[l',n]}(0)}{\partial b_{[l,i]}}$ are set to zero. Equations (161) through (165) must be applied separately for each weight and bias in the network [3]. It should be also noted that if the RMLP with GF is trained on-line, using a moving window of length $(p+1)$ as depicted in Figure 3, instead of a fixed training set of length NP, then the conditional predictions, $\hat{y}_{NN}(\cdot|t=0)$, used throughout this section must be replaced by conditional predictions of the form $\hat{y}(\cdot|t-p+1)$, to reflect the continually updated predictions using sensor readings.

III.5.4 NN Predictor Design Methodologies

The learning algorithms for the FMLP and RMLP NNs were discussed in the earlier sections. However, in practice, in addition to the learning algorithms there are many considerations that must be accounted for when training NNs for use as predictors or filters. Some of the important NN architectural design considerations, such as the data set selection, selection of the number of layers/nodes used, are discussed in this section. For a more detailed discussion, the reader is referred to [16],[65]. The important NN design considerations are as follows:

1. Data Set Selection

The data set used to train/test the NNs are the actual system measurements and/or data collected from a system-validated model. Typically, this data comprises of measurements made for a variety of different system operating conditions. This is done to ensure that the data set used in training the NN is representative of the system operations. It should be noted that the resulting NN, as with other empirical models, will be only as good as the data used to train it. Although NNs have shown to be good at generalization in a variety of application, the NN designer should still ensure that reliable and accurate data is used in the NN development. This increases the probability of success for the application the NN is intended for, whether in prediction or in state filtering.

The data set used in the development of NNs is further classified as *estimation data set* and *validation data set*. The estimation data set, also referred to as the training data set in the NN literature, is the actual data that will be used to determine the NN architecture. Again, the estimation data set comprises of two parts: the first part is the actual data used in the NN learning algorithms to modify the NN weights, and the second part is the data used to test the performance of the NN *during* the training process. This latter data set is also referred to as the *cross-validation* data set. The validation data set, on the other hand, is used to test the final NN architecture after the NN development has been completed. The NN designer must also ensure that the data sets, estimation and validation, are as different from each other as possible. The various data sets utilized are depicted in Figure 23.

During off-line learning, the distinction, between the development of the NN predictor and the NN state filter, other than the obvious differences in the inputs to the NNs, is in the *target* values used in the estimation data set. The target values are used to determine the error function to be minimized. The error functions, for the NN predictors for example, is represented by equation (142). In the development of the NN predictors, the target values are the system outputs that are to be predicted. In the development of the NN state filters, the target values are the system states that are to be filtered. In case the system states (to be filtered) cannot be measured

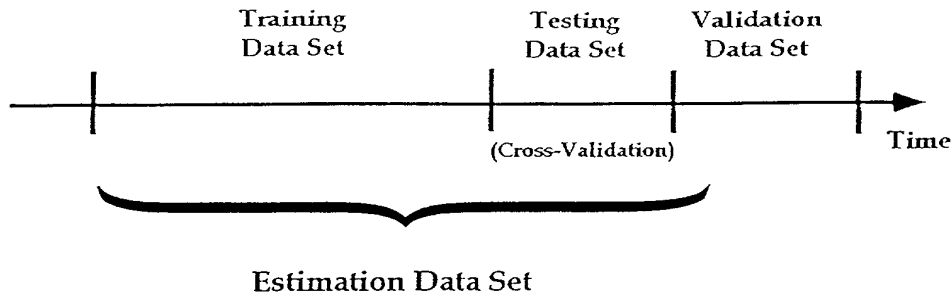


Figure 23. Diagram of the Estimation and Validation Data Sets Used to Develop a NN.

directly in off-line experiments, then they must be obtained from a reliable system model.

2. Over-fitting

Over-fitting, also referred to as over-training in the NN literature, refers to the training of a NN to the point that the NN “memorizes” the training data but it performs poorly on the validation data set. An over-fitted NN, therefore, implies that the NN is incapable of good generalization, which is an undesirable characteristic. To avoid over-fitting, the estimation data set must be carefully chosen. The data used to actually train and to test the NN during the training process must be sufficiently different from each other. In addition, the test error should be monitored during the NN training process so that if there is any indication that the test error is rising (indicative of over-fitting), the training process must be stopped.

3. Number of Layers and Nodes

The weights and bias of a NN are analogous, as mentioned in earlier sections, to the free parameters of a conventional empirical model. The number of weights and biases in a NN are dependent on the number of hidden layers and nodes in the NN. In general, following the “Occam’s Razor” principle that was popularized in the NN literature, the NN should have as few weights and biases as possible, while meeting

a specific error criterion (set by the NN designer). However, it is not clear as to how to transfer this principle to the specific task of setting the number of NN layers and nodes. While it is generally believed that only one hidden layer suffice's for most problems, the specific number of nodes is entirely dependent on the nature of the system that is being modeled by the NN. However, there are some instances where two hidden layers are found to yield better results than just one hidden layer (the number of weights and biases being approximately equal in both instances [65]).

4. Stopping Criterion

Stopping criterion is the method used to stop presentation of the estimation data to the network. It is another way by which the network approximation capability can be controlled. As the network is being trained, it passes from relatively simple to relatively complex mappings. This increase in complexity is not steady during training, but occurs in spurts. Each spurt adds a wrinkle or feature to the mapping function. Thus, the network passes through successive stages in which the number of hidden nodes that really having some positive effect goes up one by one. As the network is further trained, and its mapping function grows more complex, at some point it passes through one configuration that gives the best generalization; after that, what the network learns amounts to over-fitting. Training should be stopped at this crucial point after which any further training leads to over-fitting. The Normalized Mean-Square Error (NMSE), E_{NMSE} , is used to determine the performance of the network and is calculated as follows:

$$E_{NMSE} \equiv \frac{\sum_{t=0}^{NP-1} \sum_{j=1}^{N_{[L]}} [\hat{y}_{NN,j}(t+1|t) - y_j(t+1)]^2}{\sum_{t=0}^{NP-1} \sum_{j=1}^{N_{[L]}} y_j^2(t+1)}, \quad (167)$$

where $\hat{y}_{NN,j}(t+1|t)$ is the j -th output of the NN at time instant $t+1$ and $y_j(t+1)$ is the j -th target output at the same time instant. Hereafter, in this dissertation, unless mentioned otherwise, error refers to the NMSE defined by equation (167). In practice, an average NMSE, \overline{NMSE} , is computed when the estimation and validation

data set comprises data collected from the system (actual measurements or from a simulator) from different transients. The NMSE is first computed for each transient data set and then $\overline{\text{NMSE}}$ is computed by averaging the individual NMSEs for each transient.

5. Data Set Scaling

A simple but important step in the design of NNs is the *scaling* of the data set used in estimation/validation of the networks. The NN nodes in the hidden layers use a nonlinear squashing function, such a hyperbolic tangent, which have an output range limited to $[-1, 1]$. The effect of large inputs to the squashing functions is to cause the outputs to always be either -1 or 1 . This effect is referred to as *saturation* of the NN nodes and it is well known that excessive saturation can limit the ability of the NN to *learn* from the data set and to generalize properly. Furthermore, too large or too small data values can cause numerical problems in the calculation of the error gradients in the NN learning algorithms. For this reason, it is advantageous and sometimes necessary to limit the input and output data values to be between a narrow range. In this dissertation, all the input and output data used in estimation and validation of the NNs are restricted to be between -1 and 1 . The scaled data are generated according to the following general equation:

$$X_{\text{scaled}} = \frac{X_{\text{unscaled}} - \overline{X}_{\text{unscaled}}}{X_{\text{unscaled}}^{\max}}, \quad (168)$$

where $\overline{X}_{\text{unscaled}}$ is the average value of X_{unscaled} over N data points, and $X_{\text{unscaled}}^{\max}$ is the maximum value of X_{unscaled} . Furthermore, it should be noted that each of the inputs and outputs used in the estimation and validation should be scaled separately, because each will have different maximum values.

III.5.5 Adaptive Methods

In earlier subsections, it was assumed that a NN model used in the predictor formulation was available. In case that such a NN model, either an FMLP or an RMLP, is not available, then it must be *trained* (analogous to SI using conventional methods) using the methods developed in the preceding subsections. If, for example,

a system model is not available, then it must be identified using NNs. NNs are inherently suitable for adaptive prediction. As with other approaches, adaptive prediction methods using NNs can also be divided into *off-line* (batch and recursive) methods and *on-line* (recursive) methods.

In attempting to use NNs as adaptive predictors, either the FMLP or RMLP, either SSP or MSP, the NN structures must now be *trained*. The *learning* algorithms described in the last subsection are used for this purpose. These learning algorithms are applied, and the weights and biases of the NNs are determined from the measurement data available, while the desired prediction is performed adaptively.

III.6 Chapter Summary

In this chapter, a general framework for nonlinear prediction was developed. It was demonstrated that the general method for nonlinear prediction is based on a NARX-type system model. Nonlinear prediction methods based on conventional expansions, such as the polynomial NARX, was described. Nonadaptive methods for prediction, system identification methods, and adaptive methods for prediction (SSP and MSP), based on the NARX model, were described.

NNs, the FMLP and RMLP, were introduced in the context of alternate function expansions and were demonstrated to be of wider applicability than conventional (polynomial) expansions. The implication of this is that NNs can be used to develop higher fidelity system models for estimation, in general, and for prediction (SSP and MSP), in particular than conventional polynomial approaches. Nonadaptive methods of prediction, NN learning algorithms, and adaptive methods of prediction using NNs, both FMLP and RMLP, were described. Two NN learning algorithms, for the RMLP, with Teacher-Forcing (TF) and Global-Feedback (GF), were described in some detail. Finally, some issues related to NN predictor design were briefly described.

NONLINEAR ESTIMATION METHODS - STATE FILTERING

IV.1 Introduction

In Chapter III, nonlinear prediction methods using polynomial and neural network (NN) expansions, were discussed. In this chapter, the applicability of NNs to nonlinear estimation is taken a step further, and a framework for nonlinear state filtering using NNs, both nonadaptive and adaptive, is developed.

The standard Kalman Filter (KF) approach to state filtering, discussed in Chapter II, explicitly uses a linear system model. However, in general, most industrial systems are nonlinear and require use of nonlinear models. In one of the conventional method for performing nonlinear state filtering, that is in the Extended Kalman Filter (EKF) approach, the nonlinear system model is linearized about the state estimate and the standard KF equations are applied. The EKF suffers from convergence problems, which are partly due to the linearization of the involved nonlinear state-space model. Furthermore, in both the standard KF and the EKF, restrictive assumptions are placed on the stochastic of the process and measurement noise. Specifically, it is assumed that both the process and measurement noise are zero-mean, white, Gaussian stochastic processes. Quite often, this assumption is invalid.

As seen in Chapter III, NNs, both the Feedforward Multilayer Perceptron (FMLP) and the Recurrent Multilayer Perceptron (RMLP), are good candidates for nonlinear model structures that can be trained (identified) in a fairly systematic manner. Furthermore, NNs have been shown to be more widely applicable nonlinear model structures than conventional nonlinear model structures based on polynomial expansions. This becomes the primary motivation in using NNs in state filtering problems. Furthermore, when NNs are used in state filtering, no specific assumptions

are placed on the nature of the system nonlinearities, and as a result of the optimization problem formulated and the the direct parametrization utilized, no assumptions are placed on the statistics of the process and measurement noise. In this sense, NN state filters are comparable to the conventional nonlinear least-squares estimators, a method that also avoids the need to specify statistical models for the noise processes involved in the system model. On the contrary, minimum-variance estimators, such as the EKF, cannot avoid specifying such statistical models on the noise processes [28]. Having developed the necessary tools for using NN nonlinear predictors, the first step in the two-step prediction-update formulation of state filtering problem, this chapter now formulates and proposes a solution to the state filtering problem using NNs, both FMLPs and RMLPs.

This chapter is organized as follows: Section IV.2 describes a general framework for nonlinear state filtering. Section IV.3 discusses state filtering, using a conventional method, the EKF. Section IV.4 discusses the development of NN-based state filtering methods, and Section IV.5 summarizes the chapter.

IV.2 Nonlinear State Filtering: A General Framework

To obtain a state estimate, $\hat{\mathbf{x}}(t+1|t+1)$, using a nonlinear system model of the form given by equations (64) and (65), the following prediction-update equations can be used, which are, in principle, based on the formulation of the standard KF:

Predictor Equations:

$$\hat{\mathbf{x}}(t+1|t) = \mathbf{f}(\hat{\mathbf{x}}(t|t), \mathbf{u}(t)), \quad (169)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{h}(\hat{\mathbf{x}}(t+1|t)), \quad (170)$$

where $\hat{\mathbf{x}}(\cdot|\cdot)$ and $\hat{\mathbf{y}}(\cdot|\cdot)$ are the estimates of the system states and outputs, respectively.

Update Equation:

$$\hat{\mathbf{x}}(t+1|t+1) = \mathcal{K}(\hat{\mathbf{x}}(t+1|t), \mathcal{Y}(t+1), \mathcal{E}(t+1)), \quad (171)$$

where $\mathcal{Y}(t+1)$ is a vector containing present and past system output measurements, and $\mathcal{E}(t+1)$ is a vector containing the present and past innovations terms. These vectors are defined as follows:

$$\mathcal{Y}(t+1) \equiv [y(t+1), y(t), \dots, y(t-n_y+1)]^T, \quad (172)$$

$$\mathcal{E}(t+1) \equiv [\epsilon(t+1), \epsilon(t), \dots, \epsilon(t-n_e+1)]^T, \quad (173)$$

where, $\epsilon(t+1) \equiv y(t+1) - \hat{y}(t+1|t)$, is the innovations term defined in the standard KF, and $\mathcal{K}(\cdot)$ is a nonlinear function equivalent in functionality to the standard KF gain. Equations (169), (170) and (171) represent the general form of a nonlinear state filter, where the nonlinear functions, $f(\cdot)$, $h(\cdot)$, and $\mathcal{K}(\cdot)$ are yet to be determined. In case that a system model of the form given by equations (64) and (65) is available, then the nonlinear functions $f(\cdot)$ and $h(\cdot)$ in the filter formulation are assumed known, and $\mathcal{K}(\cdot)$ must be determined.

The error covariance matrix, $P(t+1|t+1)$, defined by equation (36) in chapter II, is a measure of the filter performance. Therefore, $P(t|t)$ can be estimated, as in the method used in the standard and the extended KF approaches, by the following predictor-update equations:

Error Covariance Predictor Equation:

The single-step predictor (SSP) of the error covariance matrix, $\hat{P}(t+1|t)$, is determined using the following equation:

$$\hat{P}(t+1|t) = \mathcal{P}(\hat{P}(t|t)), \quad (174)$$

where $\mathcal{P}(\cdot)$ is a nonlinear function.

Error Covariance Update Equation:

The SSP of the error covariance matrix, $\hat{P}(t+1|t)$, is then updated using the following equation:

$$\hat{P}(t+1|t+1) = \mathcal{Q}(\hat{P}(t+1|t), \hat{x}(t+1|t)), \quad (175)$$

where $\mathcal{Q}(\cdot)$ is another nonlinear function.

A block diagram of a generic nonlinear state filter is shown in Figure 24.

In this subsection, a general framework for nonlinear state filtering has been discussed. At this point, it is exceedingly difficult to proceed any further without restricting the nature of the nonlinearities in the models utilized, either using “analytical” assumptions or “approximation”-based arguments, leading to radically differing estimation methods. First a brief review of conventional nonlinear state filtering methods is presented, comprising of primarily the EKF. Then, the proposed formulation to NN-based state filtering is developed.

IV.3 Conventional Expansions

IV.3.1 Nonadaptive Methods

The KF, as discussed in Chapter II, is a method for system state/output filtering based on *linear* system models. If the system under investigation is described by a nonlinear state-space model, then the KF equations, (37) to (42), are not directly applicable. However, if the nonlinear state-space system model is *linearized* at a nominal estimated (conditional mean) state value (constant) or trajectory (time-varying and predetermined), then the KF equations can be applied. There are, in general, two methods used in the literature for linearizing a nonlinear model: by truncation of the involved Taylor series expansion and by statistical linearization [28]. In the following development, we adopt the former approach. The resulting state filtering method is referred to as the EKF, and it is considered a minimum-variance estimator.

As mentioned earlier, the EKF requires a system model that can be linearized. Thus, the system model is expanded in the form of a Taylor series and the higher order terms are truncated. In order that the system model be linearized in this manner, it should be differentiable, i.e. the model should be a continuous function. Consider a

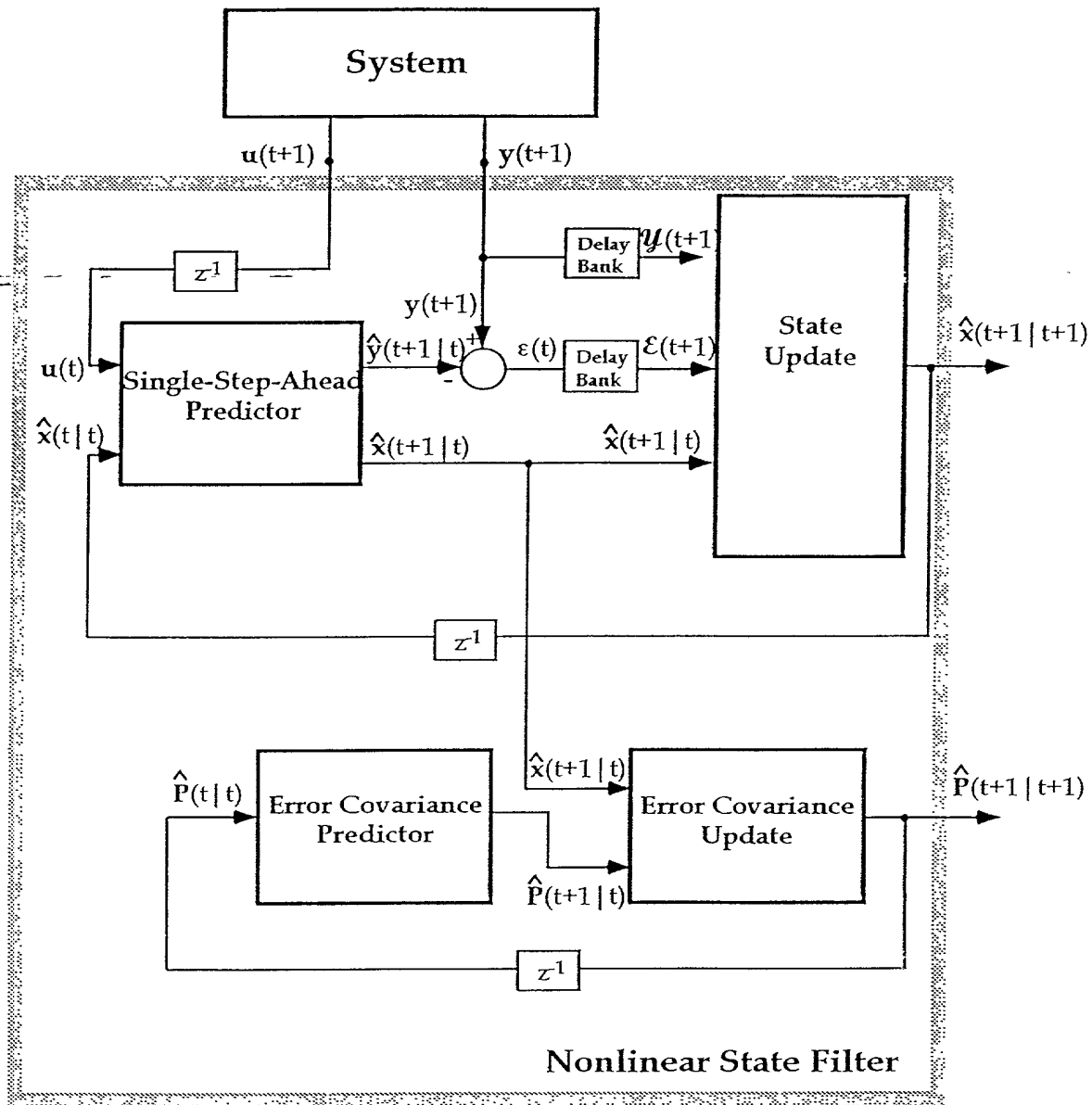


Figure 24. Block Diagram of a Nonlinear State Filter.

continuous-discrete (hybrid) nonlinear state-space system model represented by the following equations:

$$\frac{dx(\tau)}{d\tau} = f_c(x(\tau), u(\tau)) + w(\tau) \quad \text{for } t \leq \tau < t+1, \quad (176)$$

$$y(t) = h_c(x(t)) + v(t), \quad (177)$$

where $t = 1, 2, \dots$ are the time instants at which measurements are made, where $y(t)$ is the $n \times 1$ output vector; $u(t)$ is the $m \times 1$ input vector; $x(t)$ is the $l \times 1$ state vector; $f_c(\cdot)$ and $h_c(\cdot)$ are vector valued continuous nonlinear functions (known); $w(t)$, the process noise, is an $l \times 1$ vector; and $v(t)$, the measurement noise, is an $n \times 1$ vector. Furthermore, although equation (176), representing the dynamics of the states $x(\tau)$, is in continuous-time, the measurements, $y(t)$, are made at the discrete-time instants t .

The noise terms, $w(t)$ and $v(t)$, are assumed to be zero-mean, white, Gaussian vector stochastic processes with covariances Q and R , respectively, and zero cross-covariance.

The nonlinear functions $f_c(\cdot)$ and $h_c(\cdot)$ can be expanded in a Taylor series about a trajectory, $\hat{x}(t|t)$, as follows:

$$f_c(x(\tau), u(\tau)) \approx f_c(\hat{x}(t|t), u(t)) + F(\hat{x}(t|t), u(t)) [x(\tau) - \hat{x}(t|t)] \\ + G(\hat{x}(t|t), u(t)) [u(\tau) - u(t)], \quad (178)$$

$$h_c(x(\tau)) \approx h_c(\hat{x}(t|t)) + H(\hat{x}(t|t)) [x(\tau) - \hat{x}(t|t)], \quad (179)$$

where the higher order terms in the Taylor series expansions are neglected. Also, the linear vector functions $F(\hat{x}(t|t), u(t))$, $G(\hat{x}(t|t), u(t))$, and $H(\hat{x}(t|t))$ are defined as follows:

$$F(\hat{x}(t|t), u(t)) = \left. \frac{\partial f_c(x(\tau), u(\tau))}{\partial x(\tau)} \right|_{\substack{x(\tau) = \hat{x}(t|t) \\ u(\tau) = u(t)}}, \quad (180)$$

$$G(\hat{x}(t|t), u(t)) = \left. \frac{\partial f_c(x(\tau), u(\tau))}{\partial u(\tau)} \right|_{\substack{x(\tau) = \hat{x}(t|t) \\ u(\tau) = u(t)}}, \quad (181)$$

$$\mathbf{H}(\hat{\mathbf{x}}(t|t)) = \frac{\partial \mathbf{h}_c(\mathbf{x}(\tau))}{\partial \mathbf{x}(\tau)} \Big|_{\mathbf{x}(\tau) = \hat{\mathbf{x}}(t|t)} \quad (182)$$

Now, substituting the nonlinear function Taylor series expansions, represented by equations (178) and (179), in the system model represented by equations (176) and (177), the system model can be represented as follows:

$$\begin{aligned} \frac{d\mathbf{x}(\tau)}{d\tau} = & \mathbf{f}_c(\hat{\mathbf{x}}(t|t), \mathbf{u}(t)) + \mathbf{F}(\hat{\mathbf{x}}(t|t), \mathbf{u}(t)) [\mathbf{x}(\tau) - \hat{\mathbf{x}}(t|t)] \\ & + \mathbf{G}(\hat{\mathbf{x}}(t|t), \mathbf{u}(t)) [\mathbf{u}(\tau) - \mathbf{u}(t)] + \mathbf{w}(\tau) \quad \text{for } t \leq \tau < t+1 \end{aligned} \quad (183)$$

$$\mathbf{y}(t) = \mathbf{h}_c(\hat{\mathbf{x}}(t|t)) + \mathbf{H}(\hat{\mathbf{x}}(t|t)) [\mathbf{x}(\tau) - \hat{\mathbf{x}}(t|t)] + \mathbf{v}(t). \quad (184)$$

Taking the expectation of both sides of equation (183) results in the following equation [28]:

$$\frac{d\hat{\mathbf{x}}(\tau)}{d\tau} = \mathbf{f}_c(\hat{\mathbf{x}}(t|t), \mathbf{u}(t)), \quad t \leq \tau < t+1 \quad (185)$$

The KF equations can now be readily applied as discussed in Chapter II. As before, define the *Error Covariance Matrix*, $\mathbf{P}(t|t)$, as follows:

$$\mathbf{P}(t|t) = E[\mathbf{x}(t) - \hat{\mathbf{x}}(t|t)][\mathbf{x}(t) - \hat{\mathbf{x}}(t|t)]^T. \quad (186)$$

Now, referring to the timing diagram shown in Figure 25, the EKF equations are as follows:

Prior to observing the $(t+1)^{\text{th}}$ sample - Prediction Step:

The linearized nonlinear state-space model equation, (185) is used to obtain the *predictor* equation [28]:

$$\frac{d\hat{\mathbf{x}}(\tau)}{d\tau} = \mathbf{f}_c(\hat{\mathbf{x}}(\tau), \mathbf{u}(\tau)), \quad (187)$$

and the error covariance prediction equation is represented as follows:

$$\begin{aligned} \frac{d\mathbf{P}(\tau)}{d\tau} = & \mathbf{F}(\hat{\mathbf{x}}(\tau), \mathbf{u}(\tau)) \mathbf{P}(\tau) + \mathbf{P}(\tau) \mathbf{F}^T(\hat{\mathbf{x}}(\tau), \mathbf{u}(\tau)) + \mathbf{Q} \\ & - \mathbf{P}(\tau) \mathbf{H}^T(\hat{\mathbf{x}}(\tau)) \mathbf{R}^{-1} \mathbf{H}(\hat{\mathbf{x}}(\tau)) \mathbf{P}(\tau), \end{aligned} \quad (188)$$

for $t \leq \tau < t+1$. The output prediction equation is represented by the following equation:

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{h}_c(\hat{\mathbf{x}}(t+1|t)) \quad (189)$$

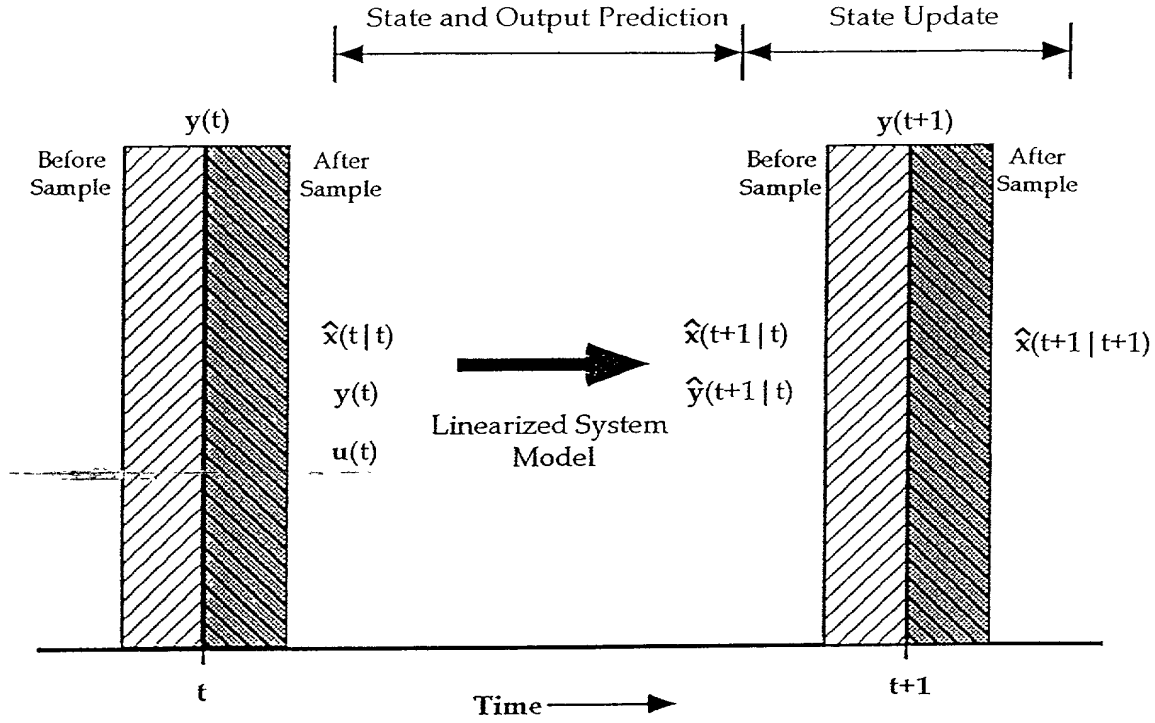


Figure 25. Timing Diagram of the Extended Kalman Filter.

Following observation of the $(t + 1)^{\text{th}}$ sample - Update Step:

After the measurement (sample) $y(t + 1)$ is obtained, the state prediction $\hat{x}(t + 1|t)$ is corrected to account for the errors in the state estimate in the prediction process [28]:

$$P(t + 1|t + 1) = \left[H(\hat{x}(t + 1|t)) R^{-1} H(\hat{x}(t + 1|t)) + P^{-1}(t + 1|t) \right]^{-1}, \quad (190)$$

$$\hat{x}(t + 1|t + 1) = \hat{x}(t + 1|t) + K_{\text{EKF}} [y(t + 1) - \hat{y}(t + 1|t)], \quad (191)$$

where

$$K_{\text{EKF}} = P(t + 1|t + 1) H(\hat{x}(t + 1|t)) R^{-1}, \quad (192)$$

where K_{EKF} is the Extended Kalman Gain Matrix. In the preceding development of the EKF equations, the state update step was *sought* to be of the form expressed by equation (191).

Because of the approximations made in linearizing the nonlinear functions, the state estimates obtained from the EKF are not usually very accurate and they exhibit convergence problems. Nevertheless, this is the only available nonlinear state filtering method. To alleviate some of the convergence problems of the EKF, the *linearized* KF is often used. In the linearized KF, the nonlinear system equations are linearized about a nominal state trajectory, $\bar{x}(t)$, instead of the state estimate $\hat{x}(t|t)$ as it was done in the EKF. The values of $\bar{x}(t)$ are determined prior to processing the measurement data.

IV.3.2 System Identification

The EKF method, as described earlier, specifically uses a nonlinear state-space system model. Usually the origin of the model is from first-principles. Therefore, the development assumes that at least the structural form of the system model is known *a priori*. If all of the parameters of the model are not known, or they are not available, then parameter estimation methods must be employed for this purpose. Unfortunately, there are no general methods for determining the parameters of such models, and one must resort to case-dependent treatment. If the nonlinear state-space model structure is itself unknown or varying, then the only available option is to re-derive the first-principles model off-line, and redesign the EKF. From the aforementioned arguments it should be clear that if the nonlinear state-space model and its associated parameters utilized in the EKF are not available, then use of the EKF becomes highly impractical, if possible at all. An alternative to the use of a state-space model might be to identify an input-output model, such as a polynomial NARX, for use with the EKF. This would be, in principle, similar to what is pursued in this research, though with polynomials instead of NNs. However, this has not been pursued in the literature.

IV.3.3 Adaptive Methods

The arguments made in the previous paragraph regarding the identification of parameters and model structures of nonlinear models, lead one to clearly suggest that

there are no general methods for adaptive state filtering using the EKF. Furthermore, specific, case-dependent, approaches to adaptive state filtering using the EKF have been scarce and not very successful. As a result these approaches have seen very limited use in real-world problems.

IV.4 Neural Network Expansions

IV.4.1 Nonadaptive Methods

In this section we develop alternative approaches to nonlinear state filtering, based on NN expansions. As with the NN predictors described in Chapter III, in addition to being nonlinear the resulting NN state filters are nonparametric in nature. Therefore, the underlying system model of these filters need not necessarily be from first-principles. Furthermore, because linearization of system nonlinearities is not involved, the derivative of any function is not utilized. As a result the entire development of the NNs state filters can be based on discrete-time, rather than hybrid, continuous-discrete, system models.

Based on the developments of Section 2, a general nonlinear state filter for a system modeled by equation (64) and (65), can be expressed using the following two-step procedure (predictor/update form):

Step 1 - Prior to observing the $(t + 1)^{th}$ sample - Prediction Step:

The predictor equations are expressed as:

$$\hat{\mathbf{x}}(t + 1|t) = \mathbf{f}(\hat{\mathbf{x}}(t|t), \mathbf{u}(t)), \quad (193)$$

$$\hat{\mathbf{y}}(t + 1|t) = \mathbf{h}(\hat{\mathbf{x}}(t + 1|t)), \quad (194)$$

where $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are vector-valued nonlinear discrete-time functions and they are specified by the system model (assumed known).

Step 2 - Following observation of the $(t + 1)^{th}$ sample - Update Step:

The update equation is represented as:

$$\hat{\mathbf{x}}(t + 1|t + 1) = \mathcal{K}\left(\hat{\mathbf{x}}(t + 1|t), \mathcal{Y}(t + 1), \mathcal{E}(t + 1)\right), \quad (195)$$

where, $\mathcal{Y}(t+1)$, and $\mathcal{E}(t+1)$ are vectors defined by equations (172), (173), respectively, and where $\mathcal{K}(\cdot)$ is the nonlinear state filter gain function used in compensating for the ever-present modeling uncertainties. In the state update equation, represented by equation (195), numerous values of past state predictions, observations, and residuals have been included. This is in contrast to linear state filters which make use of only the latest values of these variables. Such a treatment becomes necessary because of the nonlinear nature of the system model, and the lack of any system-theoretic arguments limiting the number of past information to be considered.

As is the case with all nonadaptive filters, it is assumed that a system model, $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, are available and no effort is made to estimate any part of this model. The nature and source of this models need not be specified at this point. The model could be based on first-principles, it could be semi-empirical or even fully empirical in nature. The main point of interest is that given a value of the state estimate, $\hat{\mathbf{x}}(t|t)$, and the control input, $\mathbf{u}(t)$, a SSP of the state estimate, $\hat{\mathbf{x}}(t + 1|t)$, and the output, $\hat{\mathbf{y}}(t + 1|t)$, can be computed using some number of computational steps. The second step of the filtering process, the update step, involves the (corrective) computation of the state estimate, $\hat{\mathbf{x}}(t + 1|t + 1)$, from the state prediction, and the recently received observation, along with past values of the observations.

Under ideal conditions, the state filtering problem would be set-up as some type of an optimization problem, and an attempt would be made to obtain an analytic, or a semi-analytic, solution to the filter gain function $\mathcal{K}(\cdot)$. This solution, if ever found, would involve explicit use of the functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ of the system model. However, in real-world applications there are numerous problems with such an approach. The vast majority of such problems is associated with the fact that the model $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ is not easily available. In the cases that it is available, it is in a form not easily manipulated by means of mathematical analysis, i.e. it is usually in the form of

a simulator. Furthermore, the optimization problems that would be formulated as part of a nonlinear filtering problems are extremely difficult to solve, and therefore, closed-form solutions of $\mathcal{K}(\cdot)$ are almost impossible to obtain.

The use of NNs in state filtering problems is based in the observation that the nonlinear function $\mathcal{K}(\cdot)$ of equation (171) (as well as the functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ in the adaptive case) can be approximated by any one of the NN expansions previously presented.

The state update equation (or state *filter* equation), as represented by equation (195), differs from the form of the state update equations for the standard KF, equation (41) of Section 2, in that the state estimate, $\hat{\mathbf{x}}(t+1|t+1)$, is a nonlinear function of the state prediction $\hat{\mathbf{x}}(t+1|t)$, the observation $\mathbf{y}(t+1)$, and the innovations term $\epsilon(t+1)$. In addition, it depends on an unspecified number of past values of these variables. Equation (195) is in some sense a more general filtering equation than the standard KF equation (41).

By observing the LHS and the RHS of equation (195), it is seen that the update step of the filtering process involves no advancement in the time-step. Therefore, it can be assumed that the function (mapping) $\mathcal{K}(\cdot)$ is a static one (memory-less). Nevertheless, the RHS of equation (195) will, in general, contain a large number of past values of the variables present, i.e. state prediction, observations, and innovations. Therefore, a static mapping representation of $\mathcal{K}(\cdot)$, such as one obtained using an FMLP, might be highly complex and difficult to identify because of the high dimensionality of the input vector. Thus, as it has been pointed out in previous publications [56],[57], in such circumstances a dynamic recurrent NN, such as an RMLP, can be used to capture the information carried by the past values of the variables in the RHS of equation (195), while using only their most recent values as inputs.

The nonlinear filter gain function of equation (195) can therefore be approximated as follows:

$$\mathcal{K}(\cdot) \approx \mathcal{K}_{\text{NN}}(\cdot), \quad (196)$$

and the filter update step becomes:

$$\hat{\mathbf{x}}_{\text{NN}}(t+1|t+1) = \mathcal{K}_{\text{NN}}\left(\hat{\mathbf{x}}(t+1|t), \mathcal{Y}(t+1), \mathcal{E}(t+1)\right), \quad (197)$$

where the right-hand-side (RHS) of the equation (196) is one of the two possible approximations presented in previous sub-sections, i.e. either an FMLP with Teacher Forcing (TF) or an RMLP with TF. The choice of the specific NN architecture depends highly on the complexity of the filtering problem being addressed. The notation $\hat{\mathbf{x}}_{\text{NN}}(t+1|t+1)$ denotes that the value of the state estimate is based on a NN filter. A block diagram of a nonadaptive filter based on NN expansions is shown in Figure 26.

What remains to be determined is the architectural specifications of the NN state filter $\mathcal{K}_{\text{NN}}(\cdot)$. This, however, requires (system) identification algorithms or, as widely known in the NN literature, learning algorithms. This is because the NN filter is nonparametric in nature, or as widely known in the SI literature it is a directly parametrized filter. The learning algorithms and their application to designing the filter \mathcal{K}_{NN} are treated in the following sections.

Therefore, given a system model $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ and a NN state filter $\mathcal{K}_{\text{NN}}(\cdot)$, the following steps are followed to obtain estimates of the state, $\hat{\mathbf{x}}_{\text{NN}}(t+1|t+1)$:

Step 1 - Prior to observing the $(t+1)^{\text{th}}$ sample - Prediction Step:

The state and output predictions, $\hat{\mathbf{x}}(t+1|t)$ and $\hat{\mathbf{y}}(t+1|t)$, respectively, are obtained using the following equations (from equations (193) and (194)), respectively:

$$\hat{\mathbf{x}}(t+1|t) = \mathbf{f}\left(\hat{\mathbf{x}}_{\text{NN}}(t|t), \mathbf{u}(t)\right), \quad (198)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{h}\left(\hat{\mathbf{x}}(t+1|t)\right), \quad (199)$$

Step 2 - Following observation of the $(t+1)^{\text{th}}$ sample - Update Step:

The state prediction value, $\hat{\mathbf{x}}_{\text{NN}}(t+1|t)$, is corrected by using the update equation represented by equation (197), utilizing the NN state filter $\mathcal{K}_{\text{NN}}(\cdot)$.

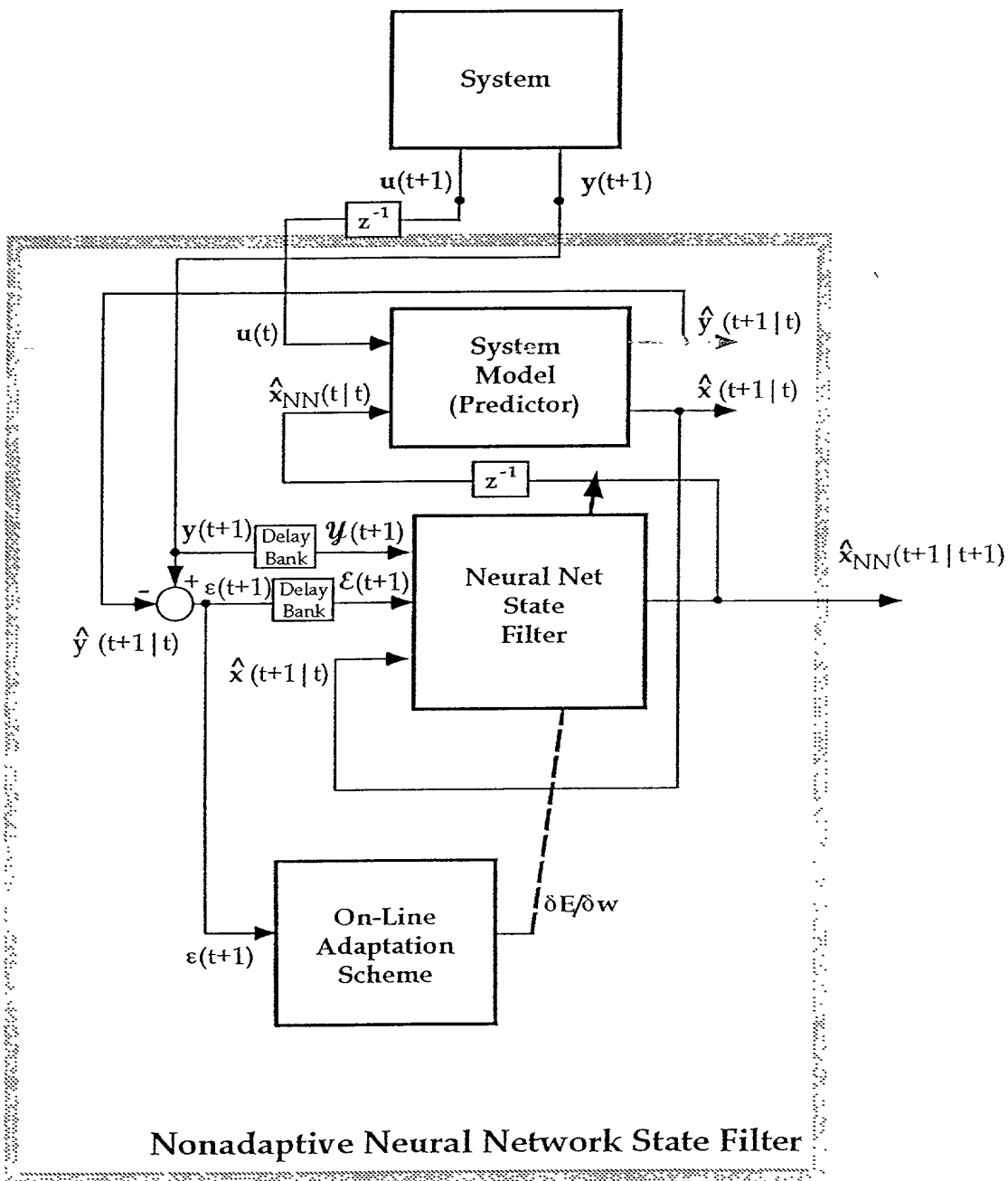


Figure 26. Block Diagram of a Nonadaptive Neural Network State Filter.

Additionally, the estimate of the error covariance, $\hat{\mathbf{P}}(t+1|t+1)$, can be obtained by using equations similar to (174) and (175). The nonlinear functions, $\mathcal{P}(\cdot)$ and $\mathcal{Q}(\cdot)$, in these equations can be approximated by NNs as follows:

$$\mathcal{P}(\cdot) \approx \mathcal{P}_{\text{NN}}(\cdot), \quad (200)$$

and

$$\mathcal{Q}(\cdot) \approx \mathcal{Q}_{\text{NN}}(\cdot), \quad (201)$$

where $\mathcal{P}_{\text{NN}}(\cdot)$ and $\mathcal{Q}_{\text{NN}}(\cdot)$ are either FMLP or RMLP NNs, with Global Feedback (GF) and TF, respectively.

Therefore, the error covariance can be estimated using the the following steps:

Step 3 - Error Covariance Prediction Step:

$$\hat{\mathbf{P}}_{\text{NN}}(t+1|t) = \mathcal{P}_{\text{NN}}(\hat{\mathbf{P}}_{\text{NN}}(t|t)), \quad (202)$$

Step 4 - Error Covariance Update Step:

$$\hat{\mathbf{P}}_{\text{NN}}(t+1|t+1) = \mathcal{Q}_{\text{NN}}(\hat{\mathbf{P}}_{\text{NN}}(t+1|t), \hat{\mathbf{x}}(t+1|t)). \quad (203)$$

Equations (198), (199), (197), (202), and (203) define a nonlinear state filter based on NN expansions. The remaining step now consists of obtaining (designing) the NNs in these aforementioned state filter equations.

IV.4.2 System Identification

NNs, as described in Chapter III, are non-parametric empirical models, and therefore they can be “re-trained” as new process measurement data becomes available. In Chapter III, the general methods used in training NNs were described in relation to their uses as predictors. Since NNs are used in filtering in a similar manner as in prediction, i.e. as approximators, the *off-line* training methods for NN state filters are algorithmically similar to the methods used in the development of the NN predictors. Specifically, if the NN state filter is an FMLP, then the BP algorithm (or

its variants) can be used. If the NN state filter is an RMLP, the learning algorithms discussed in the last chapter can be used. However, as described in the following paragraphs, in the training of the NN state filter, either the FMLP or RMLP, certain changes to the minimized error function, equation (142), must be made.

First, let us address the issues associated with off-line state filter learning. For NNs, either FMLP or RMLP with TF, the minimized error function is represented by the following equation (similar to equation (142)):

$$E \equiv \frac{1}{2} \sum_{t=0}^{NP-1} E(t+1) \equiv \frac{1}{2} \sum_{t=0}^{NP-1} \sum_{j=1}^{N_{[C]}} [\hat{x}_{NN,j}(t+1|t+1) - x_j(t+1)]^2, \quad (204)$$

where $\hat{x}_{NN,j}(t+1|t+1)$ is the j^{th} output of the NN state filter, and $x_j(t+1)$ is the corresponding state training data (target). Accordingly, the state filter output, $\hat{x}_{NN}(t+1|t+1)$, is used in place of $\hat{y}_{NN}(t+1|t)$ in the descriptions of the NN learning algorithms presented in Chapter III. Therefore, for example, the error gradients for an RMLP with TF with respect to the weights and biases can be obtained by using the chain-rule, as it was done in equations (146), (147), and (148) in Chapter III, as follows:

$$\frac{\partial E(t+1)}{\partial w_{[l,j][l,i]}} = 2 \sum_{k=1}^{N_x} [\hat{x}_{NN,k}(t+1|t+1) - x_k(t+1)] \frac{\partial \hat{x}_{NN,k}(t+1|t+1)}{\partial w_{[l,j][l,i]}}, \quad (205)$$

$$\frac{\partial E(t+1)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{k=1}^{N_x} [\hat{x}_{NN,k}(t+1|t+1) - x_k(t+1)] \frac{\partial \hat{x}_{NN,k}(t+1|t+1)}{\partial w_{[l-1,j][l,i]}}, \quad (206)$$

$$\frac{\partial E(t+1)}{\partial b_{[l,i]}} = 2 \sum_{k=1}^{N_x} [\hat{x}_{NN,k}(t+1|t+1) - x_k(t+1)] \frac{\partial \hat{x}_{NN,k}(t+1|t+1)}{\partial b_{[l,i]}}, \quad (207)$$

where N_x are the number of filtered system states.

For NNs, either the FMLP or RMLP, with GF, the minimized error function for off-line training is represented by the following equation:

$$E \equiv \frac{1}{2} \sum_{t=0}^{NP-1} E(t+1) \equiv \frac{1}{2} \sum_{t=0}^{NP-1} \sum_{j=1}^{N_{[C]}} [\hat{x}_{NN,j}(t+1|t=0) - x_j(t+1)]^2, \quad (208)$$

where $\hat{x}_{NN,j}(t+1|t=0)$ is the j^{th} output of the NN state filter, and $x_j(t+1)$ is the corresponding state training data (target). As before, using the RMLP with GF as an example, the error gradients can be represented as follows:

$$\frac{\partial E(t+1)}{\partial w_{[l,j][l,i]}} = 2 \sum_{k=1}^{N_x} [\hat{x}_{NN,k}(t+1|t=0) - x_k(t+1)] \frac{\partial \hat{x}_{NN,k}(t+1|t=0)}{\partial w_{[l,j][l,i]}}, \quad (209)$$

$$\frac{\partial E(t+1)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{k=1}^{N_x} [\hat{x}_{NN,k}(t+1|t=0) - x_k(t+1)] \frac{\partial \hat{x}_{NN,k}(t+1|t=0)}{\partial w_{[l-1,j][l,i]}}, \quad (210)$$

$$\frac{\partial E(t+1)}{\partial b_{[l,i]}} = 2 \sum_{k=1}^{N_x} [\hat{x}_{NN,k}(t+1|t=0) - x_k(t+1)] \frac{\partial \hat{x}_{NN,k}(t+1|t=0)}{\partial b_{[l,i]}}, \quad (211)$$

In the case of NN predictors discussed earlier, the targets, $y_j(t+1)$, were the system output measurements. However, in the case of the state filters, the source of targets, $x_j(t+1)$, can be either from actual state measurements obtained off-line, or from an accurate first-principles system model. Most often, accurate state measurements are not easily available and so the training data must be obtained from an accurate system model. The other considerations in NN state filter design such as overfitting, assignment of network nodes, etc., are the same as in the case of the NN predictor design, discussed in Chapter III.

On-line learning of NN, FMLP or RMLP, state filters is complicated by the fact that the only measurements available are those of the system inputs and outputs, $u(t+1)$ and $y(t+1)$, respectively. Therefore, in order to correct for deviations in the filtered state value, $\hat{x}(t+1|t+1)$, the updates for the weights and biases (parameters) must be determined from the output residual term, $e(t+1)$. Further, only a TF configuration is considered because on-line GF learning becomes exceedingly impractical. Therefore, the error function to be minimized, during on-line training, is of the form:

$$E \equiv \frac{1}{2} E(t+1) \equiv \frac{1}{2} \sum_{m=1}^{N_{[c]}} [\hat{y}_m(t+1|t) - y_m(t+1)]^2. \quad (212)$$

Therefore, the error gradients represented by equations (205), (206), and (207) will have to be modified for on-line training and can be represented as follows (again using an RMLP as an example):

$$\frac{\partial E(t+1)}{\partial w_{[l,j][l,i]}} = 2 \sum_{k=1}^{N_x} \sum_{m=1}^{N_{[c]}} \left[\hat{y}_m(t+1|t) - y_m(t+1) \right] \frac{\partial \hat{y}_m(t+1|t)}{\partial \hat{x}_{NN,k}(t|t)} \frac{\partial \hat{x}_{NN,k}(t|t)}{\partial w_{[l,j][l,i]}}, \quad (213)$$

$$\frac{\partial E(t+1)}{\partial w_{[l-1,j][l,i]}} = 2 \sum_{k=1}^{N_x} \sum_{m=1}^{N_{[c]}} \left[\hat{y}_m(t+1|t) - y_m(t+1) \right] \frac{\partial \hat{y}_m(t+1|t)}{\partial \hat{x}_{NN,k}(t|t)} \frac{\partial \hat{x}_{NN,k}(t|t)}{\partial w_{[l-1,j][l,i]}}, \quad (214)$$

$$\frac{\partial E(t+1)}{\partial b_{[l,i]}} = 2 \sum_{k=1}^{N_x} \sum_{m=1}^{N_{[c]}} \left[\hat{y}_m(t+1|t) - y_m(t+1) \right] \frac{\partial \hat{y}_m(t+1|t)}{\partial \hat{x}_{NN,k}(t|t)} \frac{\partial \hat{x}_{NN,k}(t|t)}{\partial b_{[l,i]}}. \quad (215)$$

The additional gradient in equations (213), (214), and (215), $\frac{\partial \hat{y}_m(t+1|t)}{\partial \hat{x}_{NN,k}(t|t)}$, is the gradient of the system model output with respect to the NN state filter value. The remainder of the gradients used in calculating the on-line weights and bias updates are determined in the same manner as in the off-line training methods.

In adaptive NN state filters, which are discussed in the next section, the system model is itself a NN, either FMLP or RMLP. Therefore, during on-line training, the gradient $\frac{\partial \hat{y}_m(t+1|t)}{\partial \hat{x}_{NN,k}(t|t)}$ is replaced with $\frac{\partial \hat{y}_{NN,m}(t+1|t)}{\partial \hat{x}_{NN,k}(t|t)}$. The latter gradient is the gradient of the NN system model output with respect to one of its inputs. Since the NN, either the FMLP or RMLP, are described by algebraic equations with sigmoidal functions, the computation of this gradient is straight-forward. This property is very advantageous during on-line NN state filter training.

IV.4.3 Adaptive Methods

Adaptive state filters are used when a system model is not available or if the available model is very inaccurate. The conventional method for state filtering, that is obtaining the state estimate $\hat{\mathbf{x}}(t+1|t+1)$ using the KF, requires that several assumptions be made on the system model given by equations (64) and (65). Most egregious of these assumptions is that the functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, assuming that they are known, are linear and that the noise terms $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are zero-mean, white,

Gaussian stochastic processes. In the EKF method for state filtering, the functions $f_c(\cdot)$ and $h_c(\cdot)$ are nonlinear. However, these nonlinear functions $f_c(\cdot)$ and $h_c(\cdot)$ are *linearized* about the state estimate or about a known state trajectory (in the linearized KF), and the standard KF equations are used. The noise terms are still assumed to be zero-mean, white, Gaussian stochastic processes. Thus, it is seen that there is currently no satisfactory method for obtaining state estimates in which the system model is assumed to be nonlinear, let alone unknown.

In this section, it is assumed that system model, represented by functions $f(\cdot)$ and $h(\cdot)$ in equations (64) and (65), is not available. Now, rewriting the nonlinear state-space equations (64) and (65) in its *innovations* form (similar to the linear state-space innovations form represented by equations (34) and (35)), we obtain:

$$\hat{\mathbf{x}}(t+1|t) = \mathcal{F}_{\text{innov}}(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t), \mathcal{E}(t)), \quad (216)$$

$$\mathbf{y}(t+1) = \mathcal{H}_{\text{innov}}^*(\hat{\mathbf{x}}(t+1|t)) + \mathbf{e}(t+1), \quad (217)$$

where $\mathcal{F}_{\text{innov}}(\cdot)$ and $\mathcal{H}_{\text{innov}}^*(\cdot)$ are nonlinear functions, and the other variables are as previously defined. Substituting equation (216) in equation (217) and eliminating the error term, the output prediction, $\hat{\mathbf{y}}(t+1|t)$ can be written as:

$$\hat{\mathbf{y}}(t+1|t) = \mathcal{H}_{\text{innov}}(\hat{\mathbf{x}}(t|t-1), \mathbf{u}(t), \mathcal{E}(t)), \quad (218)$$

where $\mathcal{H}_{\text{innov}}(\cdot)$ is a composite nonlinear function of the form $\mathcal{H}_{\text{innov}}^*(\mathcal{F}_{\text{innov}}(\cdot))$.

The “innovations” function, $\mathcal{E}(t)$, was defined earlier to be equal to $\mathcal{Y}(t) - \hat{\mathcal{Y}}(t|t-1)$ where the vector $\hat{\mathcal{Y}}(t|t-1)$ is defined as follows:

$$\hat{\mathcal{Y}}(t|t-1) \equiv [\hat{\mathbf{y}}(t|t-1), \hat{\mathbf{y}}(t-1|t-2), \dots, \hat{\mathbf{y}}(t-n_y+1|t-n_y)]^T, \quad (219)$$

The purpose of the “innovations” term in a nonlinear state filter, as well as in a linear state filter, is to account for stochastic effects and system modeling uncertainties unpredictable by the filter prediction step. Therefore, in the “innovation” form of the state-space equations, the state prediction, $\hat{\mathbf{x}}(t|t-1)$, is *corrected* by the innovations term, $\mathcal{E}(t)$. Now, the transition from the nonlinear “innovations” form of the state-space, as represented by equations (216) and (218), to the *prediction-update* form is

made by considering this role of the innovations term, $\mathcal{E}(t)$. It should be noted that in model-based state estimation the burden of accurate estimation rests with the system model. The update step, in the two-step prediction-update state filtering process, uses the most current system output measurements, in the form of innovations, to correct for stochastic and/or modeling inaccuracies unpredictable by the prediction step. Therefore, in the prediction step, when $\hat{\mathbf{x}}(t+1|t)$ is obtained, only the most relevant signals are required: the filtered state value, $\hat{\mathbf{x}}(t|t)$, the measured system inputs, $\mathbf{u}(t)$, and the present and/or past output predictions, $\hat{\mathbf{y}}(t|t-1)$, or output measurements, $\mathbf{y}(t)$. Since the state filter value, $\hat{\mathbf{x}}(t|t)$, has already been *corrected* by the innovations term, $\mathcal{E}(t)$, which includes the part of $\mathbf{y}(t)$ that is not included in $\hat{\mathbf{y}}(t|t-1)$, the use of both $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t|t-1)$ in obtaining the state/output prediction is redundant. Therefore, the choice is made to use $\hat{\mathbf{y}}(t|t-1)$ over $\mathbf{y}(t)$ in the prediction equations, because the latter has, potentially, a higher noise content.

From the preceding arguments, the *prediction* equations, (169) and (170), can be inferred to be represented by the following equations:

$$\hat{\mathbf{x}}(t+1|t) = \mathcal{F}^*(\hat{\mathbf{x}}(t|t), \mathbf{u}(t), \hat{\mathbf{y}}(t|t-1)), \quad (220)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathcal{H}^*(\hat{\mathbf{x}}(t|t), \mathbf{u}(t), \hat{\mathbf{y}}(t|t-1)), \quad (221)$$

where $\mathcal{F}^*(\cdot)$ and $\mathcal{H}^*(\cdot)$ are nonlinear functions.

In earlier sections it was argued that NNs are effective expansions for nonlinear functions. Furthermore, it was argued that NNs can be used to develop predictors. In view of this, the nonlinear functions $\mathcal{F}^*(\cdot)$ and $\mathcal{H}^*(\cdot)$ can be approximated by NNs. This is represented as:

$$\mathcal{F}^*(\cdot) \approx \mathcal{F}_{\text{NN}}(\cdot), \quad (222)$$

$$\mathcal{H}^*(\cdot) \approx \mathcal{H}_{\text{NN}}(\cdot), \quad (223)$$

where $\mathcal{F}_{\text{NN}}(\cdot)$ and $\mathcal{H}_{\text{NN}}(\cdot)$ are the NNs, either FMLPs or RMLPs, used to approximate the state and output predictor equations, respectively. The relationship between the

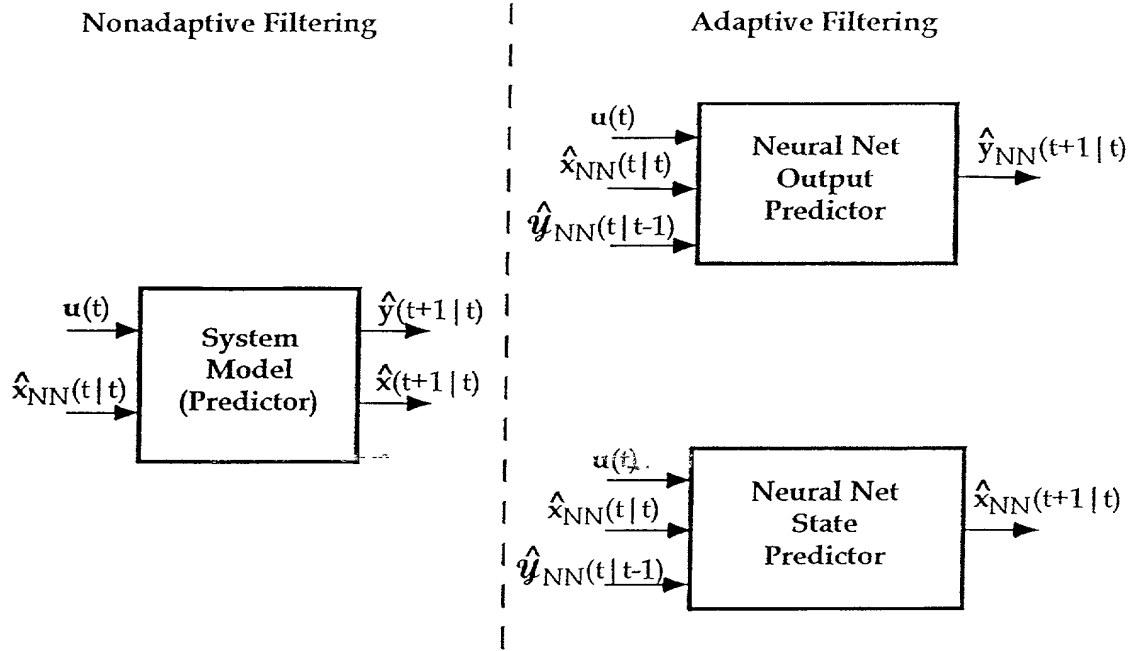


Figure 27. Block Diagram Showing the Relation between the Predictors Used in the Nonadaptive and Adaptive Neural Network State Filters.

system model predictors in the nonadaptive and adaptive methods is depicted in Figure 27.

The state update step in the nonlinear state filtering algorithm, represented by equation (171), is accomplished in a similar manner with a NN filter function $\mathcal{K}_{NN}(\cdot)$, as it was argued earlier in the nonadaptive filter section. Nevertheless, the filter input is somewhat different as discussed in the next paragraph.

The NNs, represented by the functions $\mathcal{F}_{NN}(\cdot)$, $\mathcal{H}_{NN}(\cdot)$, and $\mathcal{K}_{NN}(\cdot)$, are then designed using the learning methods discussed earlier. The state estimates, $\hat{x}(t+1|t+1)$, are then obtained using the following steps:

Step 1 - Prior to observing the $(t + 1)^{th}$ sample - Prediction Step:

The state and output predictor values are obtained using the following equations:

$$\hat{\mathbf{x}}_{\text{NN}}(t + 1|t) = \mathcal{F}_{\text{NN}}\left(\hat{\mathbf{x}}_{\text{NN}}(t|t), \mathbf{u}(t), \hat{\mathbf{y}}_{\text{NN}}(t|t - 1)\right), \quad (224)$$

$$\hat{\mathbf{y}}_{\text{NN}}(t + 1|t) = \mathcal{H}_{\text{NN}}\left(\hat{\mathbf{x}}_{\text{NN}}(t|t), \mathbf{u}(t), \hat{\mathbf{y}}_{\text{NN}}(t|t - 1)\right). \quad (225)$$

Step 2 - Following observation of the $(t + 1)^{th}$ sample - Update Step:

$$\hat{\mathbf{x}}_{\text{NN}}(t + 1|t + 1) = \overline{\mathcal{K}}_{\text{NN}}\left(\hat{\mathbf{x}}_{\text{NN}}(t + 1|t), \hat{\mathbf{y}}_{\text{NN}}(t + 1), \mathcal{E}(t + 1)\right). \quad (226)$$

The error covariance estimate, $\hat{\mathbf{P}}_{\text{NN}}(t + 1|t + 1)$, can be obtained using equations (202) and (203) as discussed in the nonadaptive filtering section. However, the input to the error covariance update step, equation (203), should be $\hat{\mathbf{x}}_{\text{NN}}(t + 1|t)$ instead of $\hat{\mathbf{x}}(t + 1|t)$.

A schematic diagram of the adaptive state filtering method is shown in Figure 28.

The adaptation methods discussed thus far have been primarily off-line. If on-line adaptation is required, i.e. if the weights of the NNs used as in the prediction and/or the update steps are to be modified, then on-line adaptation methods can easily be incorporated in the proposed state filtering method. The methods of on-line adaptation have been outlined in the previous section on SI.

IV.4.4 A Hybrid Adaptive Method for State Filtering

In the nonadaptive method of state filtering, it was assumed that the system model, represented by the functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ in equations (169) and (170), is an accurate representation of the actual system. However, if the available system model is determined to be inaccurate and if the inaccuracy is determined to be of a *deterministic* nature, then an *ad hoc* approach to state filtering can be developed. The methods for the nonadaptive and adaptive NN state filter can be combined. The resulting new approach is referred to as a *hybrid* adaptive NN state filtering method.

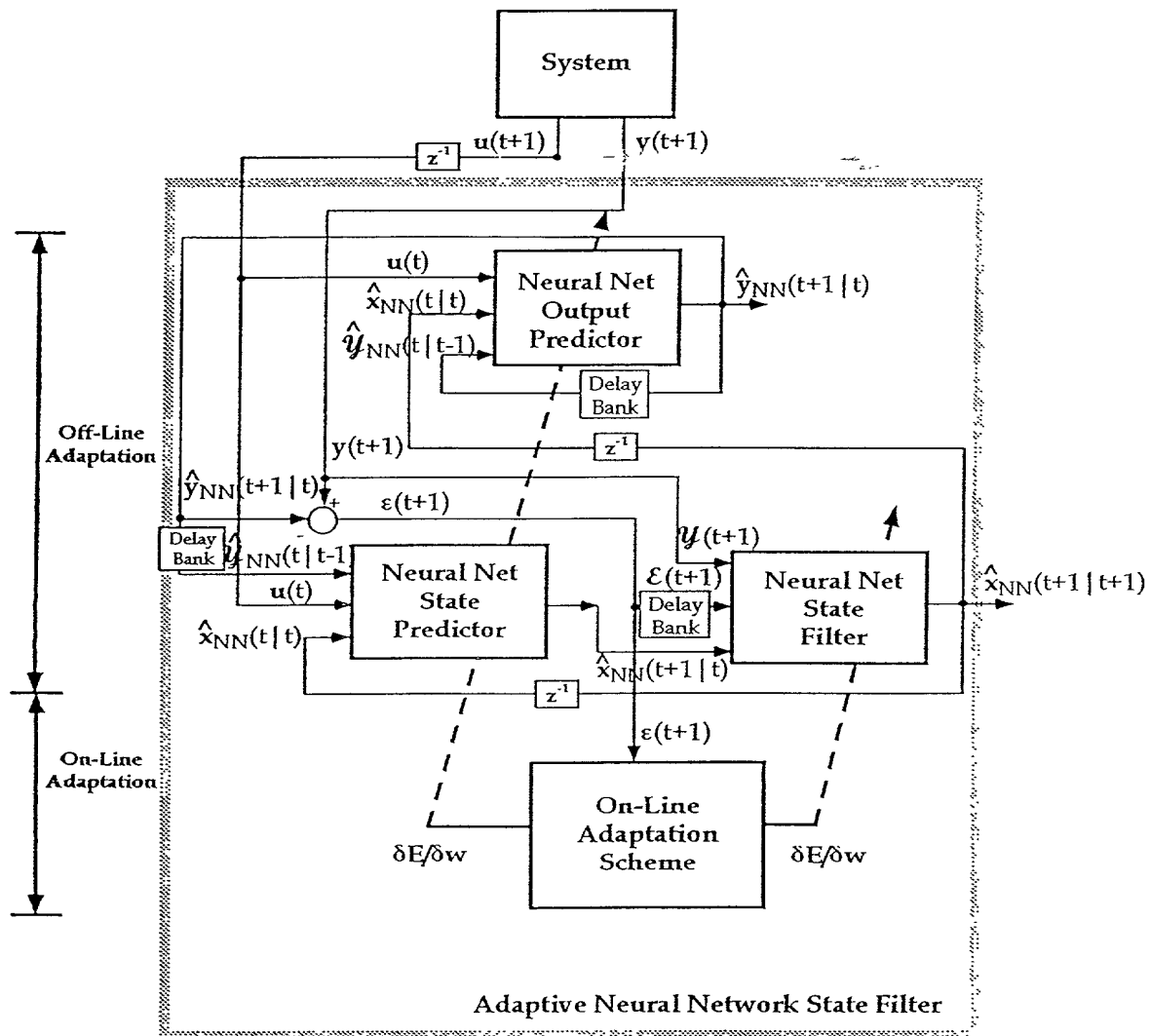


Figure 28. Block Diagram of an Adaptive Neural Network State Filter.

In this approach, the available system model is used. However, the output of the model is corrected by an additional NN, referred to as the “error model”, to account for deterministic modeling inaccuracies. Therefore, the output of the system model is corrected by adding it to the output of the “error model”. The inputs to the “error model”, at time instant $(t + 1)$ are the past inputs $\mathbf{u}(t)$, the past output measurements $\mathcal{Y}(t + 1)$, and the past past outputs of the “error model” $\hat{\mathcal{Y}}_e(t|t)$. The output of the “error model” is the *output correction* term $\hat{\mathbf{y}}_e(t + 1|t + 1)$. The error model output is expressed as:

$$\hat{\mathbf{y}}_e(t + 1|t + 1) = \mathcal{G}_{\text{NN}}(\mathbf{u}(t), \mathcal{Y}(t + 1), \hat{\mathcal{Y}}_e(t|t)), \quad (227)$$

where,

$$\hat{\mathcal{Y}}_e(t|t) \equiv [\hat{\mathbf{y}}_e(t|t), \hat{\mathbf{y}}_e(t - 1|t - 1), \dots, \hat{\mathbf{y}}_e(t - n_{ye} + 1|t - n_{ye} + 1)]^T, \quad (228)$$

where n_{ye} is the number of delays of the vector $\hat{\mathbf{y}}_e(\cdot)$, and where, $\mathcal{G}_{\text{NN}}(\cdot)$ is the function representation of the NN error model. The *corrected* system model output is expressed as:

$$\hat{\mathbf{y}}_c(t + 1|t + 1) = \hat{\mathbf{y}}(t + 1|t) + \hat{\mathbf{y}}_e(t + 1|t + 1). \quad (229)$$

The state predictor-update equations for the hybrid adaptive state filter are the same as for the nonadaptive state filtering equations (193) and (195), where the *corrected* model output $\hat{\mathbf{y}}_c(t + 1|t + 1)$ is used instead of $\hat{\mathbf{y}}(t + 1|t)$.

The error covariance estimate, $\hat{\mathbf{P}}_{\text{NN}}(t + 1|t + 1)$, can be obtained as discussed in the nonadaptive NN state filter, using equation similar to equations (202) and (203).

The block diagram of the hybrid adaptive NN filter is shown in Figure 29.

IV.5 Neural Network State Filter Performance Evaluation

The discussion thus far have dealt with the design of NN-based state filters. In order to assess the performance of the designed state filters, some form of the state estimation error must be computed. The error covariance term, $P(t)$, is such a measure of the state estimation error. Calculation of the estimate of the error covariance term, $\hat{P}(t|t)$, was discussed in earlier sections. However, in this study, estimates of the error covariance estimate matrix were not used to evaluate the performance of the NN state filters. Instead, in the case studies presented in the following chapters, two normalized errors, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are used to indicate the relative performance of the NN-based state filters. These errors are defined as follows:

$$\epsilon_{\text{Sim}}(t) = \frac{\mathbf{x}(t) - \hat{\mathbf{x}}_{\text{NN}}(t|t)}{\mathbf{x}(t)}, \quad (230)$$

$$\epsilon_{\text{Mod}}(t) = \frac{\mathbf{x}_m(t) - \hat{\mathbf{x}}_{\text{NN}}(t|t)}{\mathbf{x}_m(t)}. \quad (231)$$

The normalized error, $\epsilon_{\text{Sim}}(t)$, is a measure of the accuracy of the state estimate, $\hat{\mathbf{x}}_{\text{NN}}(t|t)$, with respect to the “true” state value, $\mathbf{x}(t)$, as obtained from the system simulator (or the actual system itself if available). However, another measure of the performance of the state filter is $\epsilon_{\text{Mod}}(t)$. The normalized error $\epsilon_{\text{Mod}}(t)$ is a measure of the accuracy of the state estimate, $\hat{\mathbf{x}}_{\text{NN}}(t|t)$, with respect to the state value, $\mathbf{x}_m(t)$, as determined by the system model. Since $\mathbf{x}_m(t)$ is used to develop the state filter, and in the absence of on-line updating of the the NN filter, $\hat{\mathbf{x}}_{\text{NN}}(t|t)$ will approach the system model state, $\mathbf{x}_m(t)$. Consequently, modelling errors determine how close $\mathbf{x}_m(t)$, and consequently $\hat{\mathbf{x}}_{\text{NN}}(t|t)$, is to the actual (or simulator) system state, $\mathbf{x}(t)$. In this research, the average $\epsilon_{\text{Sim}}(t)$, $\overline{\epsilon_{\text{Sim}}(t)}$, and the average $\epsilon_{\text{Mod}}(t)$, $\overline{\epsilon_{\text{Mod}}(t)}$, are also computed for the duration of the transient.

It should be noted, however, that for actual implementation of a NN state filter, whether nonadaptive or adaptive, and whether the adaptation is performed off-line or on-line, some type of an error covariance matrix estimate will be necessary.

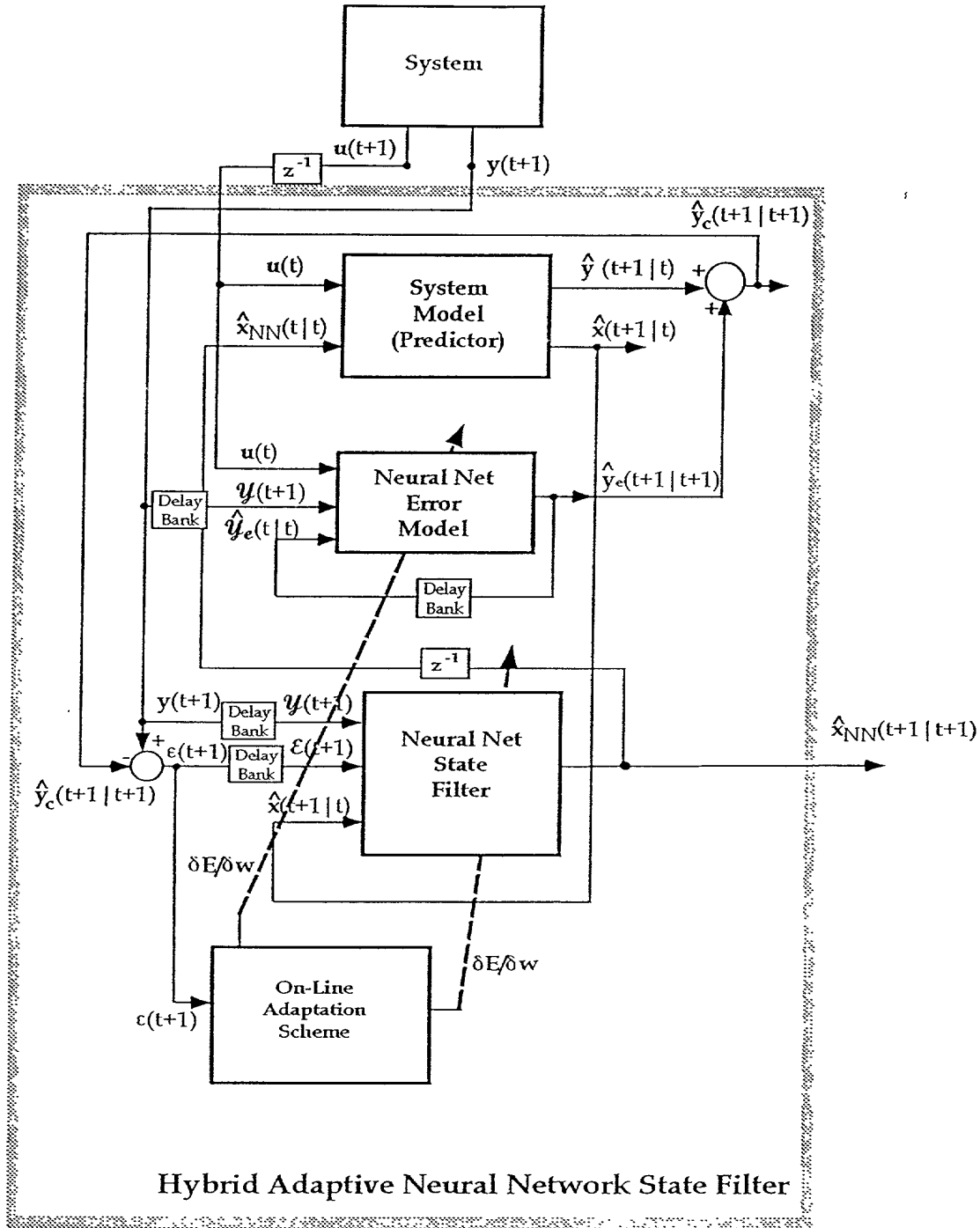


Figure 29. Block Diagram of a Hybrid Adaptive Neural Network State Filter.

IV.6 Chapter Summary

This chapter discussed a general framework for nonlinear state filtering. A nonadaptive method for state filtering using conventional methods, such as the EKF, was presented. A NN state filtering method which uses an available process model was then discussed. It was seen that the NN state filtering approaches are of far more general applicability than conventional approaches, because of the effectiveness of NNs as accurate nonlinear function approximators. Nonlinear SI was then discussed with regards to NN-based filters, and some differences in the learning between NN predictors and NN state filters learning were presented. An adaptive method for state filtering using NNs was then presented. The adaptive method is based on NN predictors, which replace of the first-principles process model in the nonadaptive method, and NN state filters. Finally, a hybrid adaptive method for NN state filtering is presented. The hybrid adaptive method is similar to the nonadaptive method, with the exception that in the former a NN “error model” is added to compensate for deterministic inaccuracies in the process model output.

APPLICATION 1 - TWO-INPUT-TWO-OUTPUT SYSTEM

V.1 Introduction

This chapter presents the first of three case-studies applying the NN-based state filtering methods discussed in Chapter IV. A stable Two-Input-Two-Output (2I2O) nonlinear dynamic system is chosen to apply the NN state filtering methods. The chosen 2I2O system, although with a relatively simple structure, is nonlinear and exhibits a sufficiently nontrivial behavior to apply and test the NN state filtering methods. In particular, a nonadaptive and an adaptive NN state filter is developed for this 2I2O system and tested via simulations.

The chapter is organized into 5 sections: Section V.2 provides a detailed description of the 2I2O system. Section V.3 describes the models of this 2I2O system used in the development of the NN state filters. Section V.4 describes the development of the nonadaptive and adaptive NN state filters for the 2I2O system. Section V.5 presents the results of test evaluations performed on the developed NN state filters. The final section, V.6, summarizes the chapter.

V.2 System Description

The 2I2O stochastic nonlinear system used in this study is described by the following state and output equations:

$$x_1(t+1) = 0.5 \left(x_1(t) \right)^{\frac{2}{3}} + 0.3 x_2(t) x_3(t) + 0.2 u_1(t) + n_{x_1}^{\text{process}}(t), \quad (232)$$

$$x_2(t+1) = 0.5 \left(x_2(t) \right)^{\frac{2}{3}} + 0.3 x_3(t) x_1(t) + 0.5 u_1(t) + n_{x_2}^{\text{process}}(t), \quad (233)$$

$$x_3(t+1) = 0.5 \left(x_3(t) \right)^{\frac{2}{3}} + 0.3 x_1(t) x_2(t) + 0.5 u_2(t) + n_{x_3}^{\text{process}}(t), \quad (234)$$

and

$$y_1(t+1) = 0.7 \left(x_1(t+1) + x_2(t+1) \right), \quad (235)$$

$$y_2(t+1) = 1.5 x_1^2(t+1), \quad (236)$$

respectively, where $n_{x_i}^{\text{process}}(t)$ for $i = 1, 2, 3$, is the zero-mean, white, Gaussian process noise added to the system.

This is a 2I2O system with three dynamic states, $x_1(t)$, $x_2(t)$, and $x_3(t)$, corrupted by process noise. The two outputs are $y_1(t)$ and $y_2(t)$, and the two inputs are $u_1(t)$ and $u_2(t)$.

The state to be estimated is $x_3(t)$. It should be noted that only two of the states, $x_1(t)$ and $x_2(t)$, are explicitly present in the equations describing the outputs, $y_1(t)$ and $y_2(t)$. This implies that the state to be estimated affects the 2I2O measured system outputs only indirectly.

V.3 System Model Description

Two system models, Model 1 and Model 2, are used to simulate the 2I2O system and thereby provide approximate values of the state $x_3(t)$. These approximate values, $x_{m3}(t)$, are then used to train the empirical (NN) models.

Model 1 is an accurate representation of the 2I2O system; Model 2 is a less accurate representation of the same system. While the differences in the constant coefficients between the two models is small, these differences are enough to cause significant variations in the model outputs. This is because the model outputs are extremely sensitive to changes in the numerical values of the constant coefficients. These models are discussed in greater detail in the following subsections.

V.3.1 System Model 1 - An Accurate Analytical Model

The state and output equations describing Model 1 are given by (the subscript m refers to the model):

$$x_{m1}(t+1) = 0.5 \left(x_{m1}(t) \right)^{\frac{3}{5}} + 0.3 x_{m2}(t) x_{m3}(t) + 0.2 u_1(t), \quad (237)$$

$$x_{m2}(t+1) = 0.5 \left(x_{m2}(t) \right)^{\frac{3}{5}} + 0.25 x_{m3}(t) x_{m1}(t) + 0.5 u_1(t), \quad (238)$$

$$x_{m3}(t+1) = 0.5 \left(x_{m3}(t) \right)^{\frac{3}{5}} + 0.25 x_{m1}(t) x_{m2}(t) + 0.5 u_2(t), \quad (239)$$

and

$$y_{m1}(t+1) = 0.7 \left(x_{m1}(t+1) + x_{m2}(t+1) \right), \quad (240)$$

$$y_{m2}(t+1) = 1.5 x_{m1}^2(t+1), \quad (241)$$

respectively. The *predictor* part of the *prediction-update* equations for Model 1 can be expressed as (rewriting equations (237),(238),(239),(240),(241)):

$$\hat{x}_1(t+1|t) = 0.5 \left(\hat{x}_1(t|t) \right)^{\frac{3}{5}} + 0.3 \hat{x}_2(t|t) \hat{x}_3(t|t) + 0.2 u_1(t), \quad (242)$$

$$\hat{x}_2(t+1|t) = 0.5 \left(\hat{x}_2(t|t) \right)^{\frac{3}{5}} + 0.25 \hat{x}_3(t|t) \hat{x}_1(t|t) + 0.5 u_1(t), \quad (243)$$

$$\hat{x}_3(t+1|t) = 0.5 \left(\hat{x}_3(t|t) \right)^{\frac{3}{5}} + 0.25 \hat{x}_1(t|t) \hat{x}_2(t|t) + 0.5 u_2(t), \quad (244)$$

and

$$\hat{y}_1(t+1|t) = 0.7 \left(\hat{x}_1(t+1|t) + \hat{x}_2(t+1|t) \right), \quad (245)$$

$$\hat{y}_2(t+1|t) = 1.5 \hat{x}_1^2(t+1|t). \quad (246)$$

V.3.2 System Model 2 - An Inaccurate System Model

The state and output equations describing Model 2 are given by:

$$x_{m1}(t+1) = 0.45 \left(x_{m1}(t) \right)^{\frac{3}{5}} + 0.3 x_{m2}(t) x_{m3}(t) + 0.2 u_1(t), \quad (247)$$

$$x_{m2}(t+1) = 0.45 \left(x_{m2}(t) \right)^{\frac{3}{5}} + 0.25 x_{m3}(t) x_{m1}(t) + 0.5 u_1(t), \quad (248)$$

$$x_{m3}(t+1) = 0.45 \left(x_{m3}(t) \right)^{\frac{3}{5}} + 0.25 x_{m1}(t) x_{m2}(t) + 0.5 u_2(t), \quad (249)$$

and

$$y_{m1}(t+1) = 0.7 \left(x_{m1}(t+1) + x_{m2}(t+1) \right), \quad (250)$$

$$y_{m2}(t+1) = 1.5 x_{m1}^2(t+1), \quad (251)$$

respectively. The equations (247),(248),(249),(250) and (251) can be rewritten in the *predictor* form as:

$$\hat{x}_1(t+1|t) = 0.45 \left(\hat{x}_1(t|t) \right)^{\frac{3}{5}} + 0.3 \hat{x}_2(t|t) \hat{x}_3(t|t) + 0.2 u_1(t), \quad (252)$$

$$\hat{x}_2(t+1|t) = 0.45 \left(\hat{x}_2(t|t) \right)^{\frac{3}{5}} + 0.25 \hat{x}_3(t|t) \hat{x}_1(t|t) + 0.5 u_1(t), \quad (253)$$

$$\hat{x}_3(t+1|t) = 0.45 \left(\hat{x}_3(t|t) \right)^{\frac{3}{5}} + 0.25 \hat{x}_1(t|t) \hat{x}_2(t|t) + 0.5 u_2(t), \quad (254)$$

and

$$\hat{y}_1(t+1|t) = 0.7 \left(\hat{x}_1(t+1|t) + \hat{x}_2(t+1|t) \right), \quad (255)$$

$$\hat{y}_2(t+1|t) = 1.5 \hat{x}_1^2(t+1|t). \quad (256)$$

V.4 Neural Network State Filters

This section outlines the development of the nonadaptive and adaptive NN state filters. The nonadaptive filter uses the highly accurate analytical model, Model 1, and the adaptive filter uses the less accurate analytical model, Model 2.

The design of both the nonadaptive and adaptive NN state filters requires an estimation data set and a validation data set. The estimation and validation data sets are a collection of step and ramped-sinusoids, that cover a wide range of typical inputs to which the 2I2O system is subjected. The inputs, $u_1(t)$ and $u_2(t)$, of the estimation data set are listed in Table 1. The estimation data set is generated using Model 1 for the nonadaptive filter, and Model 2 for the adaptive filter. In addition, process noise is added to the equations while data is collected. The process noise is assumed to be zero-mean, white, Gaussian with 0.01 standard deviation (sd). The total number of samples in the estimation data is 1000. Out of these 1000 samples, the NN weights are updated in only the first 600 samples, whereas the remaining 400 samples are used to evaluate the performance of the NN models (cross-validation).

Table 1. 2I2O System Estimation Data Set Used by the Nonadaptive and Adaptive NN State Filters.

Estimation Data Set			
Weight Update	$u_1(t)$	$u_2(t)$	Number of Samples
Yes	Combinations of Steps of Magnitudes: [0., 0.125, 0.25, 0.375, 0.5]		400 (20 samples each)
	$0.1 + 0.1 \sin(\pi t/10) + 0.3 (t/200)$	$0.1 + 0.3 (t/200)$	200
No	$0.3, \quad t \leq 30s$ $0.1, \quad t > 30s$	$0.3, \quad t \in [10s, 50s]$	100
	$0.005 t$	0.2	100
	0.2	$0.005 t$	100
	$0.1 + 0.1 \sin(\pi t/20) + 0.2 (t/200)$	$0.1 + 0.2 (t/200)$	100

In the following subsections, the development of the nonadaptive and adaptive NN state filters for the 2I2O system is presented in detail. All training/simulations of the NNs were performed on an SGI Power Challenge XLTM computer.

V.4.1 Nonadaptive State Filter

The nonadaptive state filtering method was described in Chapter IV. The NN state filter is first chosen to be a three-layer FMLP with 5 nodes in the first layer (corresponding to 5 inputs) and 1 node in the third layer (corresponding to 1 output). The 5 inputs in the first layer are the two system outputs, $y_1(t+1)$ and $y_2(t+1)$, the two residuals, $e_1(t+1)$ and $e_2(t+1)$, where $e_1(t+1) = y_1(t+1) - \hat{y}_1(t+1|t)$

and $\epsilon_2(t+1) = y_2(t+1) - \hat{y}_2(t+1|t)$, and the state $x_{m3}(t)$. The single output is the filtered state value, $\hat{x}_3(t+1|t+1)$. The number of nodes in the hidden layer is initially set to a value of 6. The NN state filter is then trained with the estimation data set. The target is the state $x_{m3}(t+1)$ as provided by Model 1.

Training is stopped when the NMSE of the estimation data set (without weight update) just begins to increase (indicating the start of over-training). The NMSE is noted, the number of nodes in the hidden layer is increased, and the cycle is repeated. After a sufficient number of NN architectures have been developed in this way, the best among them is chosen. This choice is based on the lowest NMSE achieved, with the lowest number of nodes. In this case, it is determined that the FMLP NN with an architecture of 5-6-1 is the best model. A block diagram of the nonadaptive filter is shown in Figure 30.

A comparison between the 2I2O system simulator outputs, $y_1(t+1)$ and $y_2(t+1)$, and the outputs from Model 1, $\hat{y}_1(t+1|t)$ and $\hat{y}_2(t+1|t)$, is shown in Figure 31. In Figure 32, the 5-6-1 FMLP NN state filter output, $\hat{x}_3(t+1|t+1)$, is shown in comparison to the Model 1 state, $x_{m3}(t+1)$, and the 2I2O Simulator state, $x_3(t+1)$ with inputs from the estimation data set.

The training procedure is repeated with a three-layer RMLP network. The best RMLP NN has an architecture of 5-6-1. However, the total estimation data set (without weight update) average NMSE for the FMLP network was 0.285%, whereas the corresponding average NMSE for the RMLP network was 3.04%. The FMLP 5-6-1 is chosen as the NN state filter based on this significant difference between the average NMSEs of the FMLP and RMLP networks.

V.4.2 Adaptive State Filter

The adaptive method using NN state filters was described in Chapter IV. Three NNs are used in this state filter: the NN output predictor, the NN state predictor and the NN state filter. The NN output predictor is developed first. The NN state predictor is then developed, with the NN output predictor running in parallel. Finally, the NN state filter is developed with both the NN output predictor and the NN state

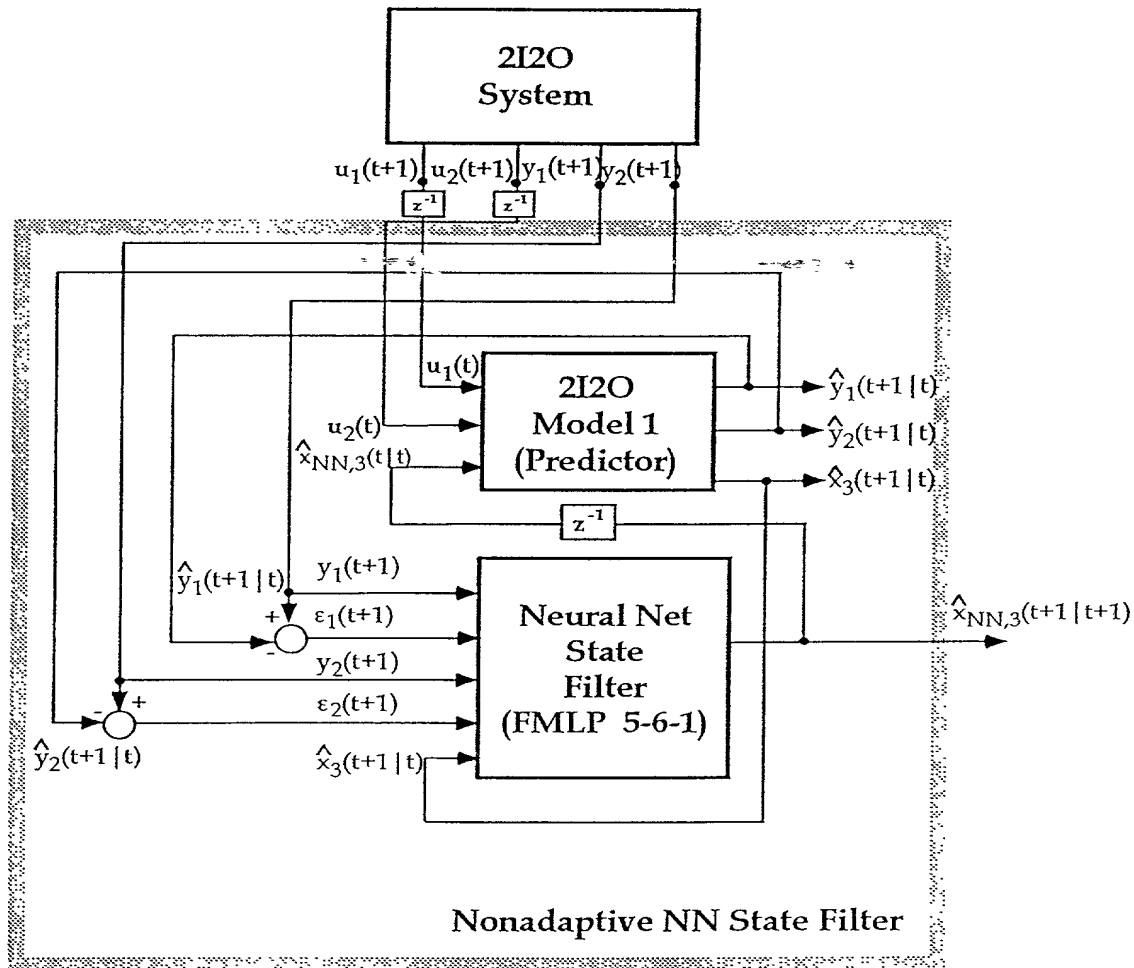


Figure 30. Block Diagram of the 2I2O System Nonadaptive NN State Filter.

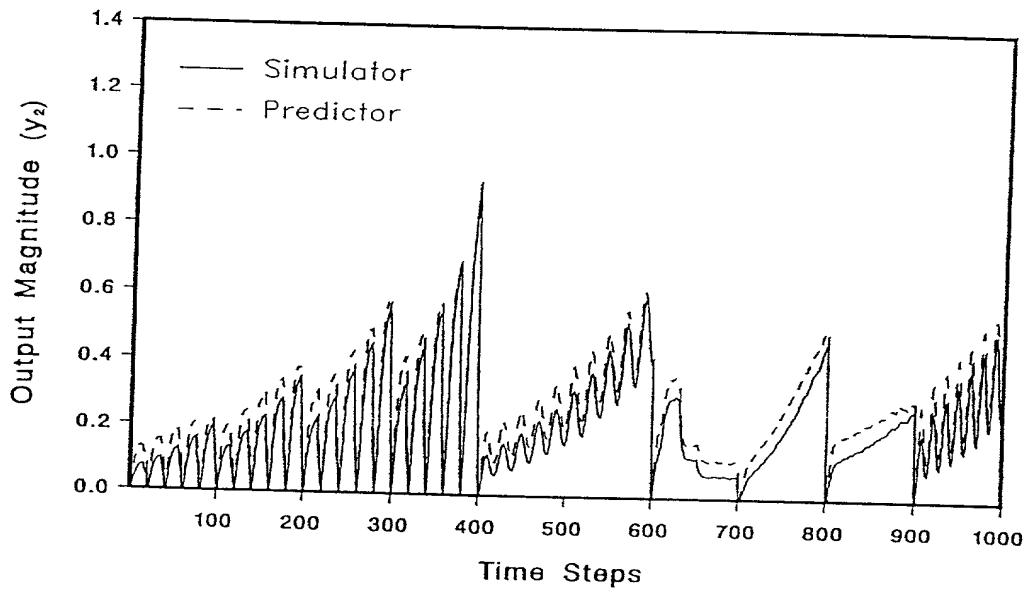
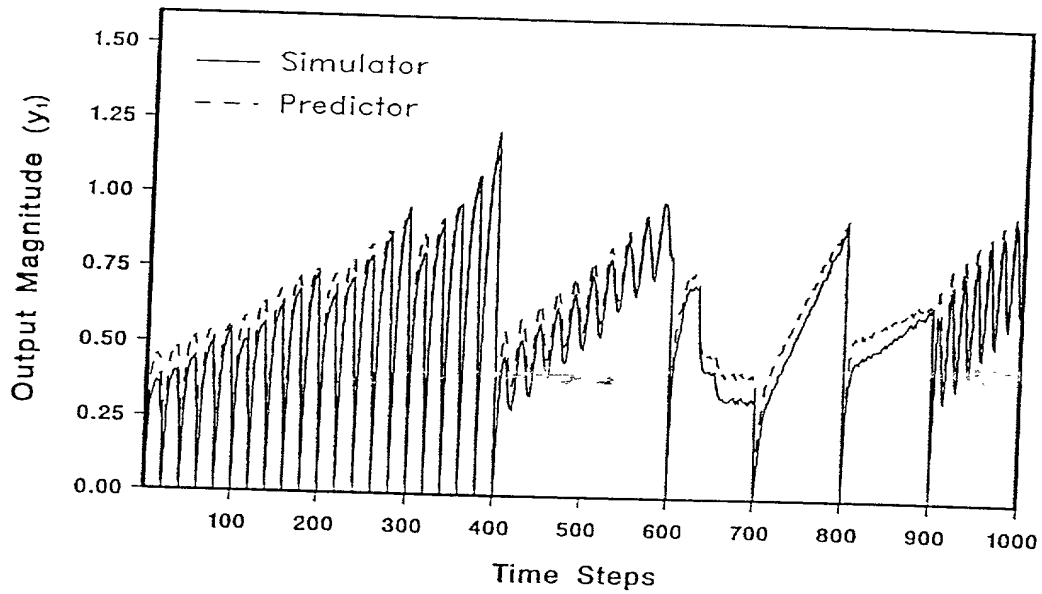


Figure 31. 2I2O System Output Response, from the Simulator and System Model 1 Predictor, for the Nonadaptive NN State Filter Using the Estimation Data Set as Inputs.

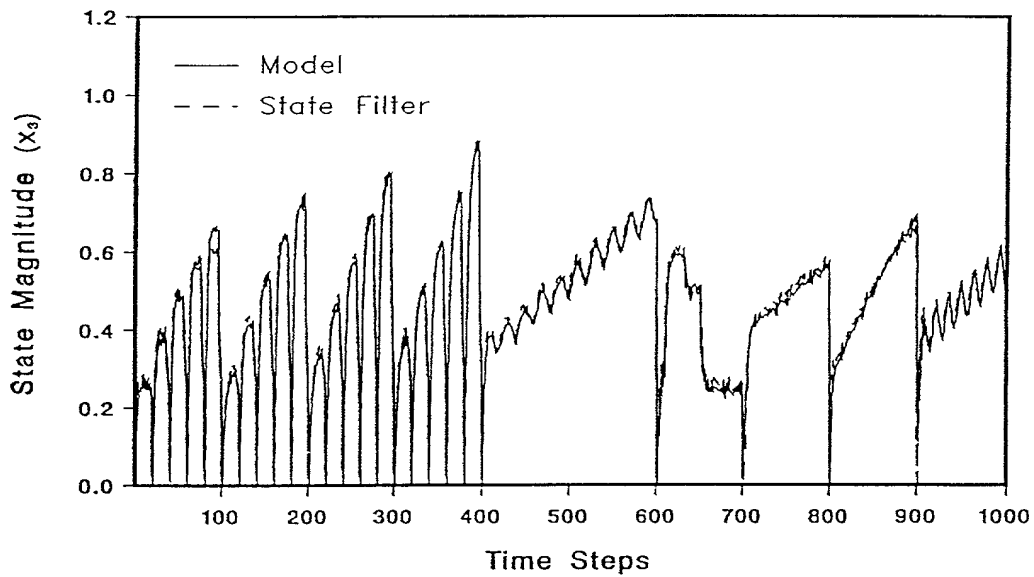
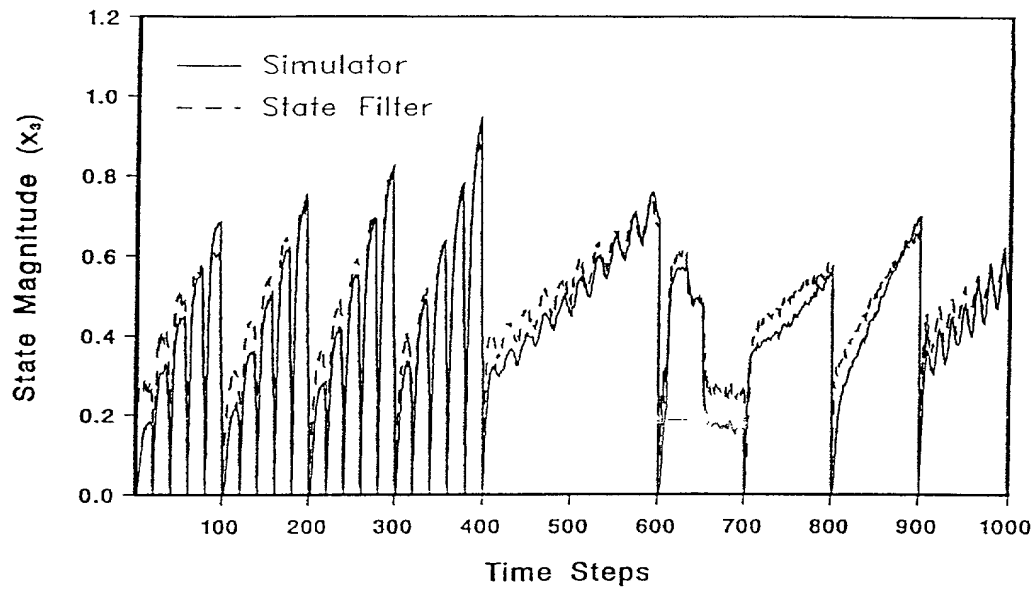


Figure 32. 2I2O System State $x_3(t)$ Response, from the Simulator, Model 1, and NN State Filter, for the Nonadaptive NN State Filter Using the Estimation Data Set as Inputs.

predictor running concurrently. The development of each of these NNs is discussed in greater detail in the following subsections.

V.4.2.1 NN Output Predictor Development

The NN output predictor is initially chosen to be a three-layer FMLP network with 5 nodes in the first layer (corresponding to 5 inputs) and 2 nodes in the third layer (corresponding to 2 outputs). The two outputs of the third layer are the SSPs of the system outputs, $\hat{y}_1(t+1|t)$ and $\hat{y}_2(t+1|t)$. The first 3 inputs in the first layer correspond to the two system inputs, $u_1(t)$ and $u_2(t)$, and the state $x_{m3}(t)$ (from Model 2). The last two inputs to the first layer are the delayed outputs from the NN output predictor, $\hat{y}_1(t|t-1)$ and $\hat{y}_2(t|t-1)$. The number of nodes in the hidden layer is initially set to 6. The NN output predictor is then trained with the estimation data set, using the system model outputs, $y_{m1}(t+1)$ and $y_{m2}(t+1)$, as targets.

Training of the NN output predictor is continued until the stopping criterion is reached; that is, until the NMSE of the estimation data set (without weight update) just begins to increase. The number of nodes in the hidden layer is increased and the cycle repeated. A suitable number of NN models is developed and the best among them is chosen. In this case, the best NN output predictor architecture was 5-6-2.

The training procedure is repeated with a three-layer RMLP network. The best RMLP network was found to be one with an architecture of 5-7-2. However, the average test data set NMSE for the best FMLP network (5-6-2) was 0.183%, whereas the corresponding average NMSE for the RMLP network (5-7-2) was 16.6%. Therefore, the FMLP network is chosen as the NN output predictor because of the order of magnitude difference in the average NMSEs between the FMLP and the RMLP networks. In Figure 33, a comparison between the system outputs, $y_1(t+1)$

and $y_2(t+1)$, and the NN output predictor outputs, $\hat{y}_1(t+1|t)$ and $\hat{y}_2(t+1|t)$, using the estimation data set as inputs is shown.

V.4.2.2 NN State Predictor Development

The NN output predictor weights are now fixed and no further training is performed on this model. The NN state predictor is chosen to be a three-layer FMLP with 5 nodes in the first layer (corresponding to 5 inputs) and 1 node in the third layer (corresponding to one output). The output of the third layer is the SSP of the state, $\hat{x}_3(t+1|t)$. The inputs to the first layer are the two outputs from the NN output predictor, $\hat{y}_1(t+1|t)$ and $\hat{y}_2(t+1|t)$, the two system inputs, $u_1(t)$ and $u_2(t)$, and the delayed output of the NN state predictor, $\hat{x}_3(t|t-1)$. The number of nodes in the hidden layer is initially set to 6. The NN state predictor is then trained with the estimation data set, using the state $x_{m3}(t)$ provided by Model 2 as the target.

The NN state predictor is trained until the NMSE of the estimation data set (without weight update) just begins to increase at which time the training is stopped. The number of nodes in the hidden layer is increased and the training procedure is repeated. A number of NN predictors is developed in this way and the best among them is chosen based on the lowest NMSE and lowest number of nodes. In this case, it was determined that the FMLP NN with an architecture of 5-8-1 performs the best. The average NMSE in the test data set was 0.612%. The RMLP NN was not used to develop models for the NN state predictor in this case, as it was determined that the performance of the FMLP NN is more than sufficient. In this case, the additional complexity introduced by the RMLP networks was not justified.

V.4.2.3 NN State Filter Development

Now, the weights of the NN output predictor and the NN state predictor are fixed and no further training is performed on them. The NN state filter architecture is initially chosen to be a three-layer FMLP NN with 5 nodes in the first layer (corresponding to 5 inputs) and 1 node in the third layer (corresponding to one output).

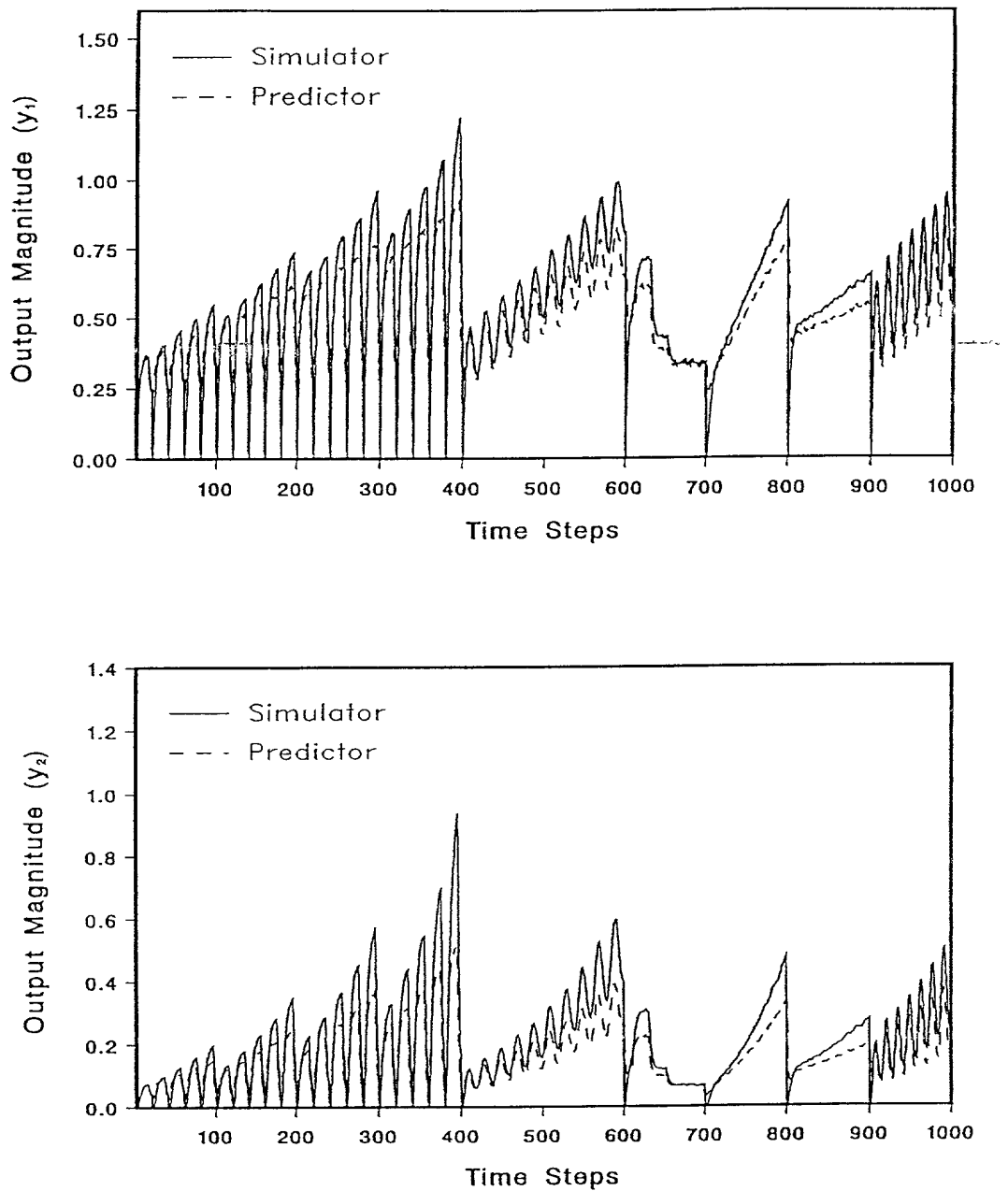


Figure 33. 2I2O System Output Response, from the Simulator and the NN Output, for the Adaptive NN State Filter Using the Estimation Data Set as Inputs.

The output of the third layer is the state SSP, $\hat{x}_3(t+1|t+1)$. The five inputs to the first layer are the two system outputs, $y_1(t+1)$ and $y_2(t+1)$, the two residuals, $\epsilon_1(t+1)$ and $\epsilon_2(t+1)$ where $\epsilon_1(t+1) = y_1(t+1) - \hat{y}_1(t+1|t)$ and $\epsilon_2(t+1) = y_2(t+1) - \hat{y}_2(t+1|t)$, and the output from the NN state predictor, $\hat{x}_3(t+1|t)$. The target used in training is the same as in the training of the NN state predictor, i.e. the state $x_{m3}(t+1)$.

The initial number of nodes in the hidden layer is set at 6 and training is performed as described in the previous section. Different NN architectures (with different number of hidden nodes) are developed. The best FMLP NN architecture was found to be a 5-6-1 network with the average NMSE on the test data set 0.213%. As with the NN state predictor, the RMLP network was not used, as the performance of the FMLP networks was more than satisfactory. The complete block diagram for the adaptive NN state filter is shown in the block diagram in Figure 34. The performance of the adaptive NN state filter using the estimation data set as inputs is shown in Figure 35.

V.5 Validation Results

The nonadaptive and adaptive NN state filters developed in the previous sections are evaluated in terms their performance on a validation data set. The validation data set comprises signals entirely different from the signals in the estimation data set used in the development of the NNs. The input signals, $u_1(t)$ and $u_2(t)$, of the validation data set are listed in Table 2.

Each test signal in the validation data set is augmented with zero-mean, white, Gaussian noise with 0.01 sd (low noise). The performance of the state filters are then evaluated. To evaluate the independent test signals under higher noise conditions, the validation data set signals are then augmented with zero-mean, white, Gaussian noise with 0.05 sd (high noise), and the performance of the nonadaptive and adaptive NN-based state filters is then evaluated.

The performance of the nonadaptive and adaptive NN state filters is described in the following subsections.

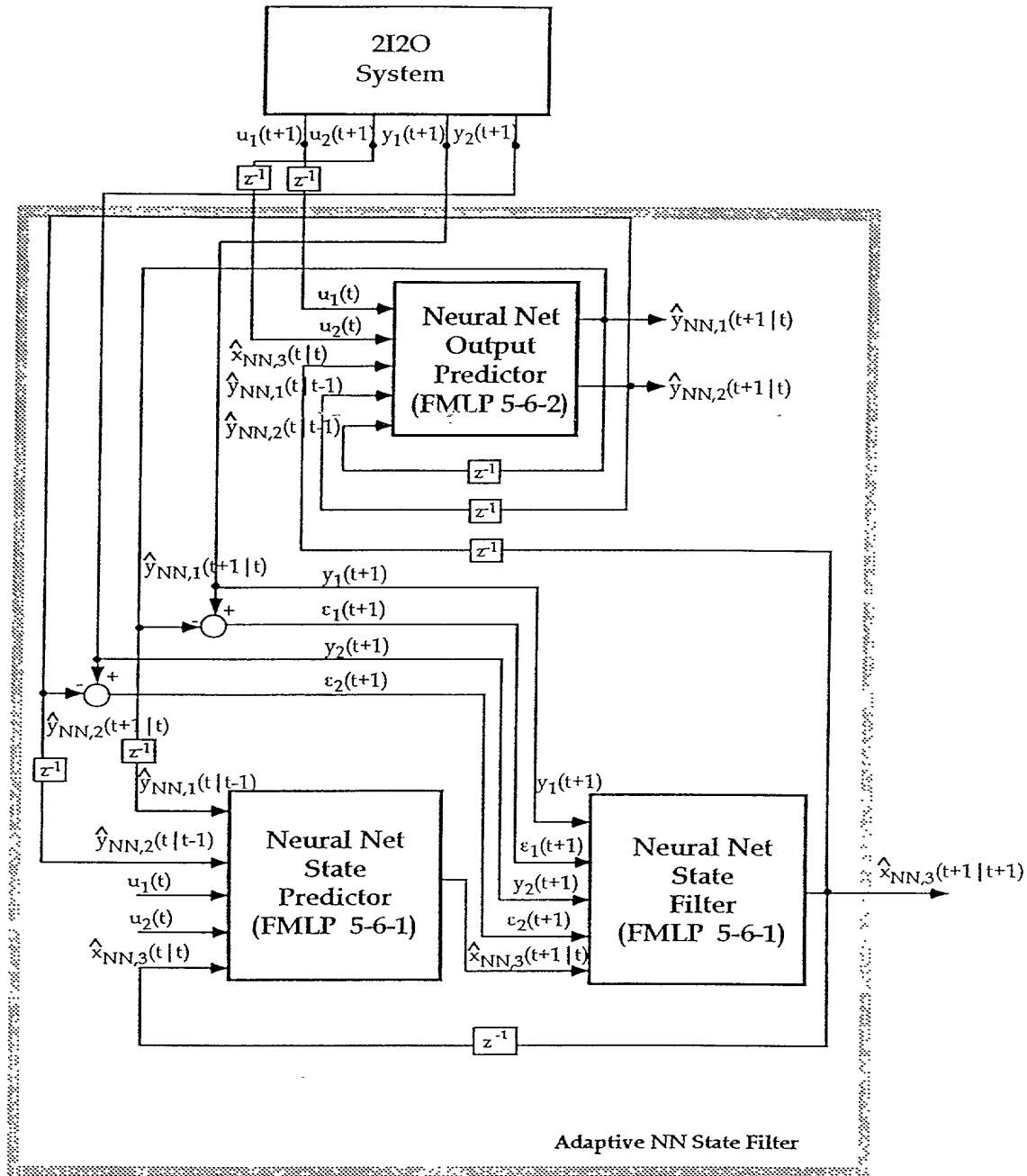


Figure 34. Block Diagram of the 2I2O System Adaptive NN State Filter.

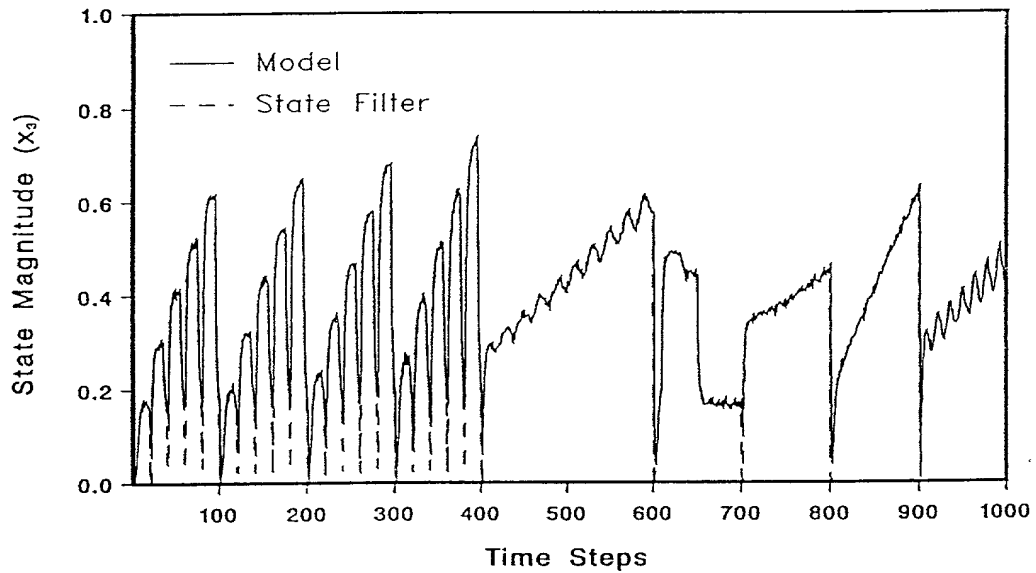
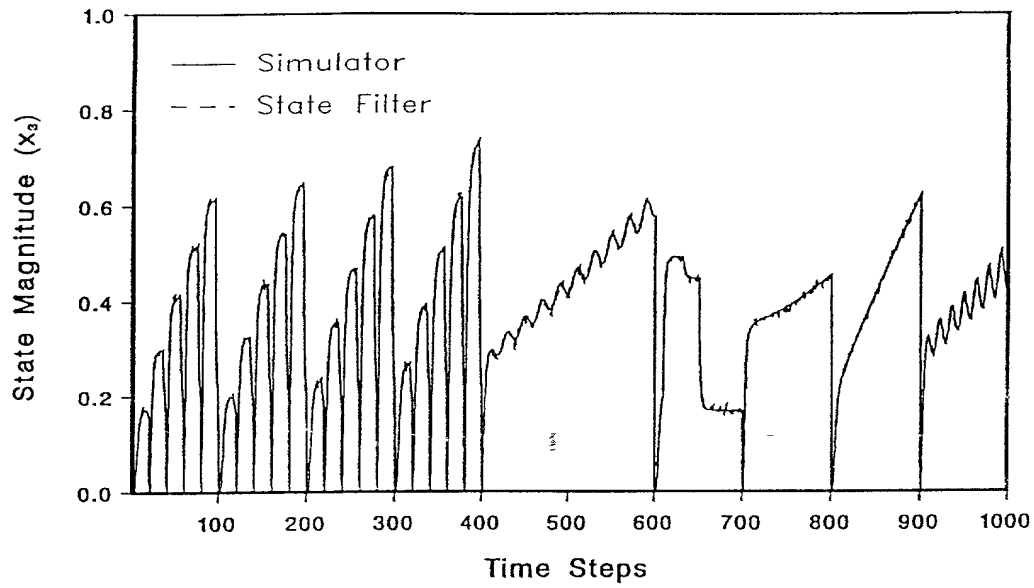


Figure 35. 2I2O System State $x_3(t)$ Response, from the Simulator, Model 2, and NN State Filter, for the Adaptive NN State Filter Using the Estimation Data Set as Inputs.

Table 2. 2I2O System Validation Data Set Used by the Nonadaptive and Adaptive NN State Filters.

Validation Data Set		
$u_1(t)$	$u_2(t)$	Number of Samples
$0.3 \sin(\pi t/8) + 0.5 (t/400)$	$0.45 (t/400) - 0.2 \sin(\pi t/20)$	100
$0.3, \quad t \leq 26s$ $0, \quad t > 26s$	$0.3, \quad t \leq 36s$ $0, \quad t > 36s$	50
$0.002 t$	0.2	50
0.2	$0.002 t$	50
$0.3 + 0.2 \sin(\pi t/5)$	$0.2 + 0.3 (t/200)$	200

V.5.1 Nonadaptive State Filter

The output transient response of the 2I2O Predictor is shown in Figure 36. The average NMSE of the filtered state value, $\hat{x}_3(t|t)$, is 0.32%. The state filter values for the validation data set is shown in Figure 37.

It should be noted from Figure 37 that the state filter value, $\hat{x}_3(t|t)$, will always show a discrepancy compared to the actual system state value, $x_3(t)$. This is because the NN state filter was trained with the Model 1 state values, $x_{m3}(t)$, as well as Model 1 inputs and outputs, and *not* with the actual system state values, $x_3(t)$, and inputs and outputs (the actual system state values, in practical situations, cannot be easily evaluated). Therefore, the filter values, $\hat{x}_3(t|t)$, will only be as accurate as the Model 1 state values, $x_{m3}(t)$. This is depicted in Figure 38 where the *normalized residuals* (errors), $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are depicted.

From Figure 38, it is seen that although the $\epsilon_{\text{Sim}}(t)$ values are low, the $\epsilon_{\text{Mod}}(t)$ values are even lower. The average $\epsilon_{\text{Sim}}(t)$ value is -12.5% and the average $\epsilon_{\text{Mod}}(t)$ value

466

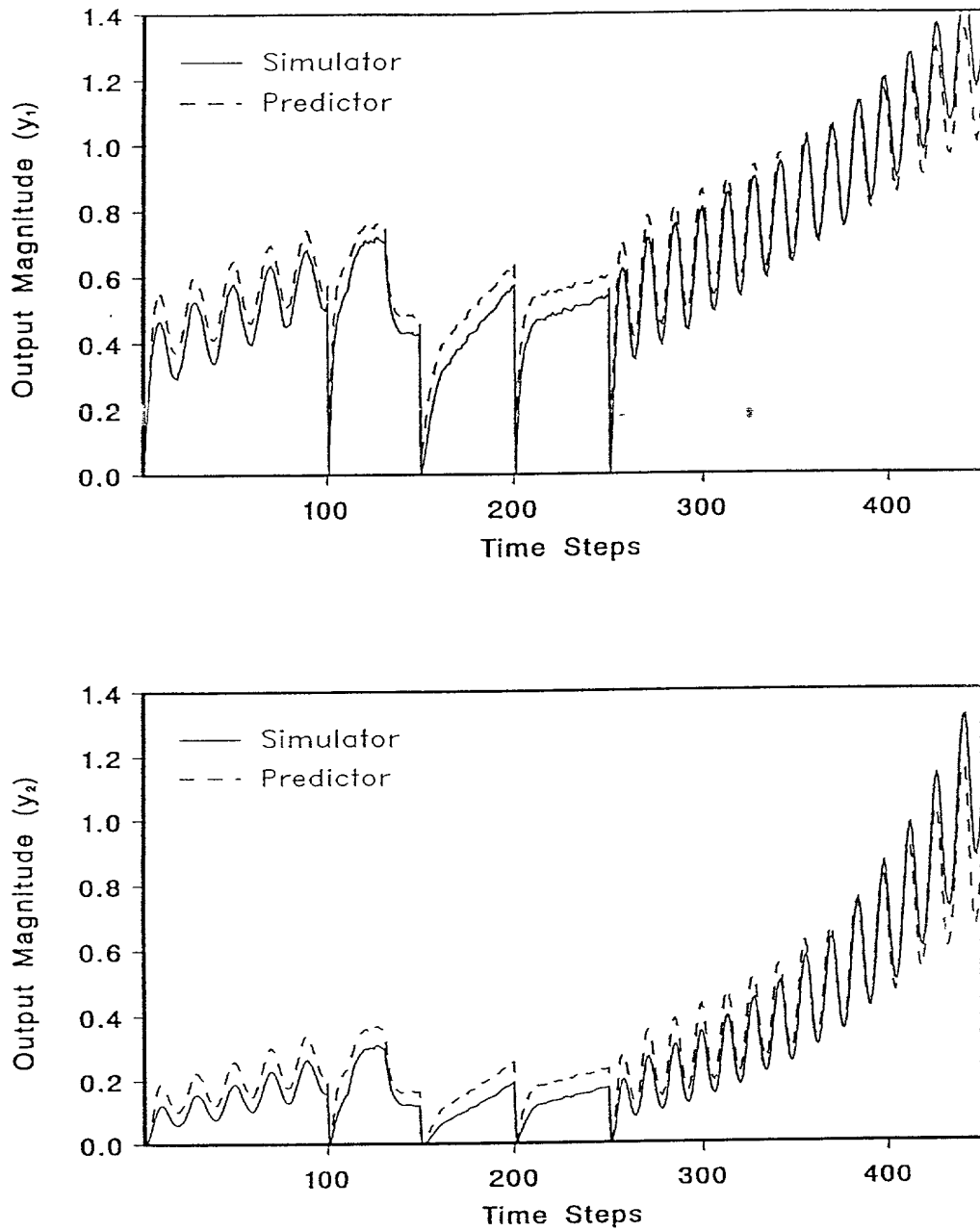


Figure 36. 2I2O System Output Response, from the Simulator and the Model Predictor, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).

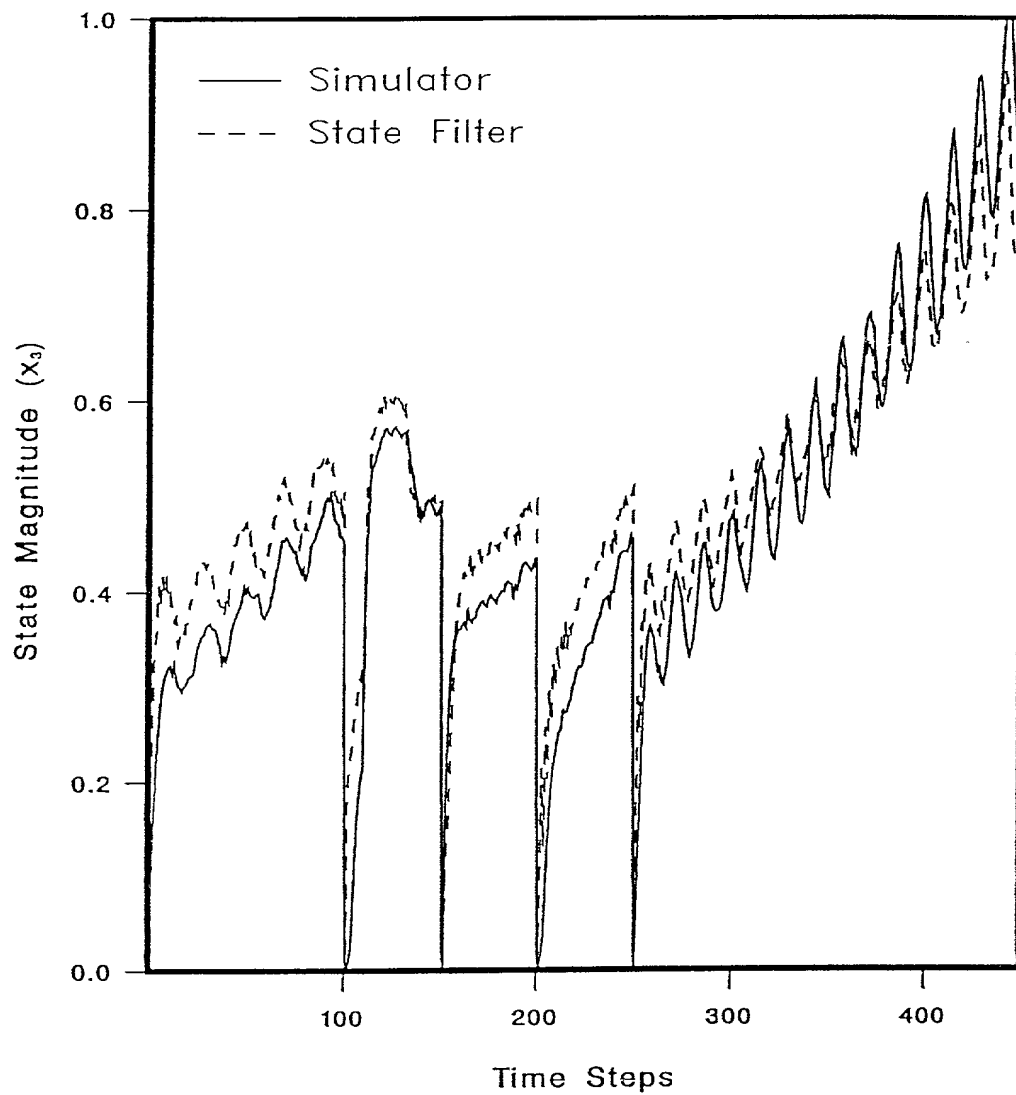


Figure 37. 2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).

468

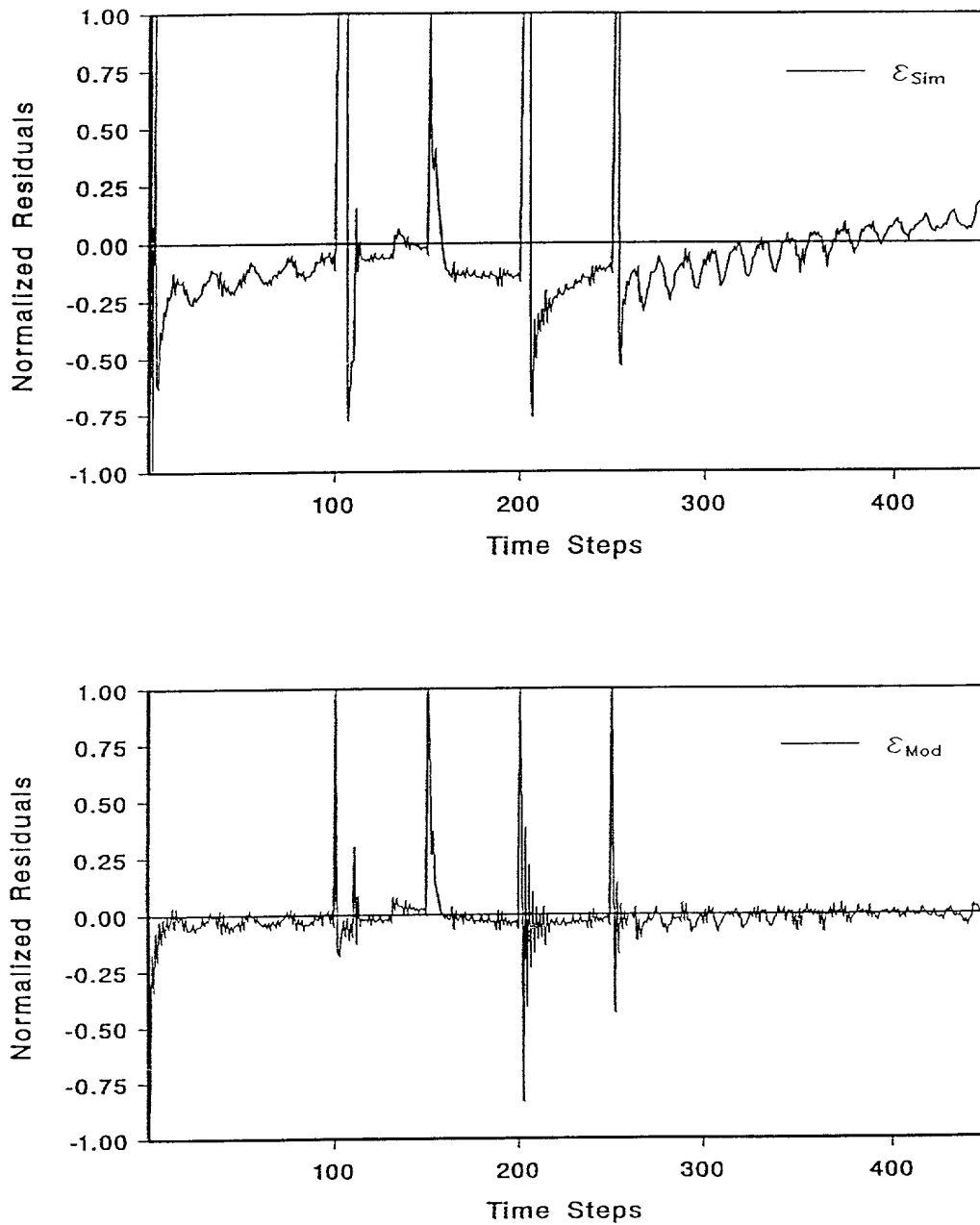


Figure 38. 2I2O State Filter Normalized Residuals for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).

is -0.1%. This is a direct consequence of the fact that the NN state filter is trained using the Model 1 state values, $x_{m3}(t)$, inputs and outputs, and not using the actual system state values, $x_3(t)$, inputs and outputs. Any further improvements to the nonadaptive state filter estimates must come from the on-line adaptation of the filter NN using actual system observations.

The performance of the nonadaptive state filter, with the same validation data set as used in the previous example, is evaluated with a higher level of process noise: zero-mean, white, Gaussian, with 0.05 sd. The output transient response of the 2I2O Predictor is shown in Figure 39. The average NMSE error of the state estimate, $\hat{x}_3(t|t)$, is 0.46%. The state estimates for the test data set with high noise are shown in Figure 40. It is seen in Figure 40 that, in spite of the presence of high noise in the test data set, the nonadaptive NN state filter provides an accurate state estimate. This is further seen in Figure 41, from the plots of the normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$. The average value of $\epsilon_{\text{Sim}}(t)$ is -11% and the average value of $\epsilon_{\text{Mod}}(t)$ is 0.05%.

V.5.2 Adaptive State Filter

The adaptive state filter is tested using the same validation data set as was used in the testing of the nonadaptive state filter, following a similar testing procedure as in the case the nonadaptive filter. Initially, zero-mean, white, Gaussian noise of 0.01 sd is added to the validation data set. The output transient response of the NN output predictor is shown in Figure 42. The average NMSE for the filtered state value, $\hat{x}_3(t|t)$, is 0.12%. The filtered state values for the validation data set are shown in Figure 43. The NN state predictor and NN state filter in the adaptive filter were trained using the Model 2 state values, $x_{m3}(t)$, inputs and outputs. Hence, the filtered states, $\hat{x}_3(t|t)$, will only be as accurate as the Model 2 state values, $x_{m3}(t)$, inputs and outputs. This is shown in Figure 44, where $\epsilon_{\text{Sim}}(t)$ is, in general, larger than $\epsilon_{\text{Mod}}(t)$.

The performance of the adaptive state filter is then evaluated using the validation data set with a higher level of process noise (zero-mean, white, Gaussian noise

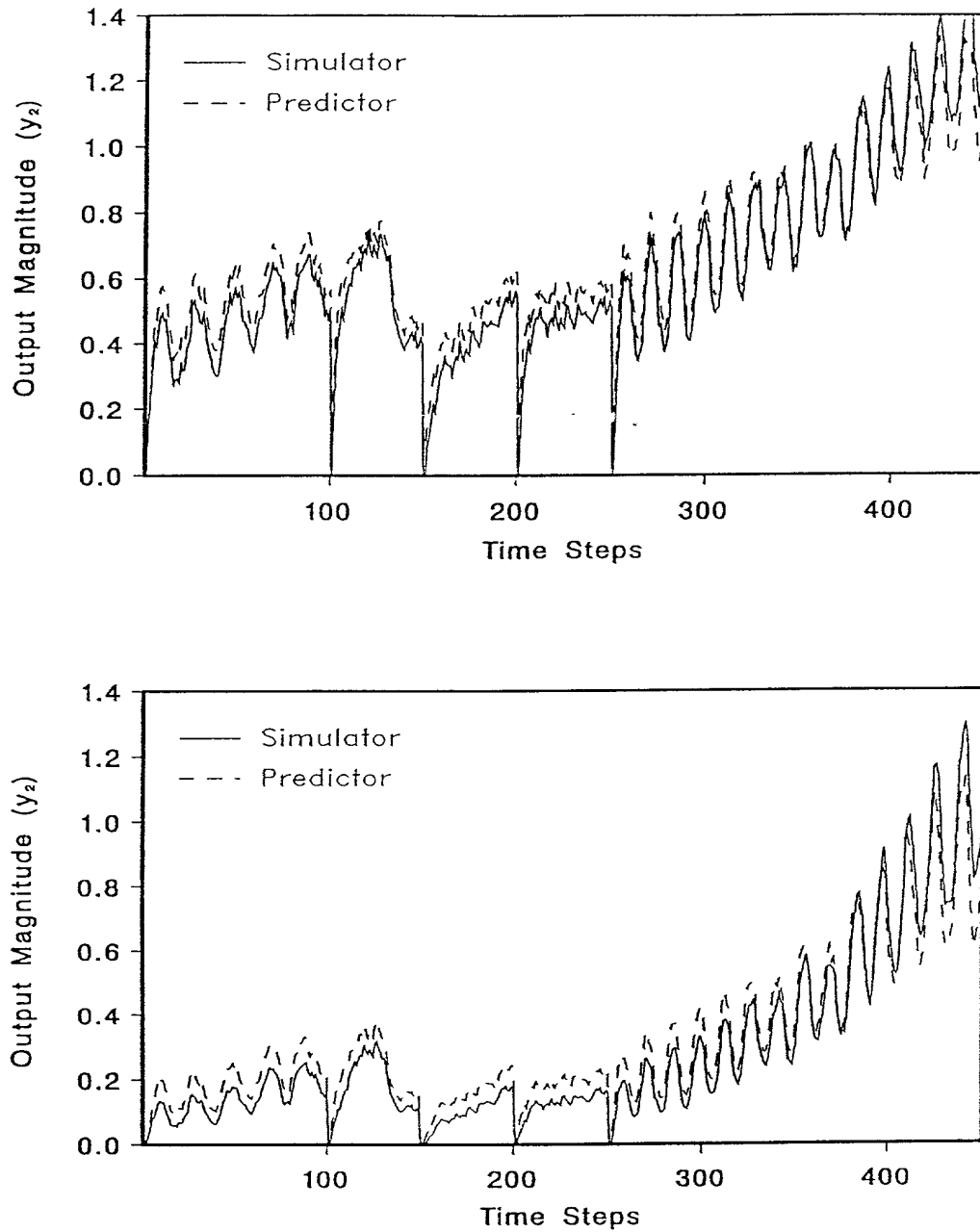


Figure 39. 2I2O System Output Response, from the Simulator and the Model Predictor, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).

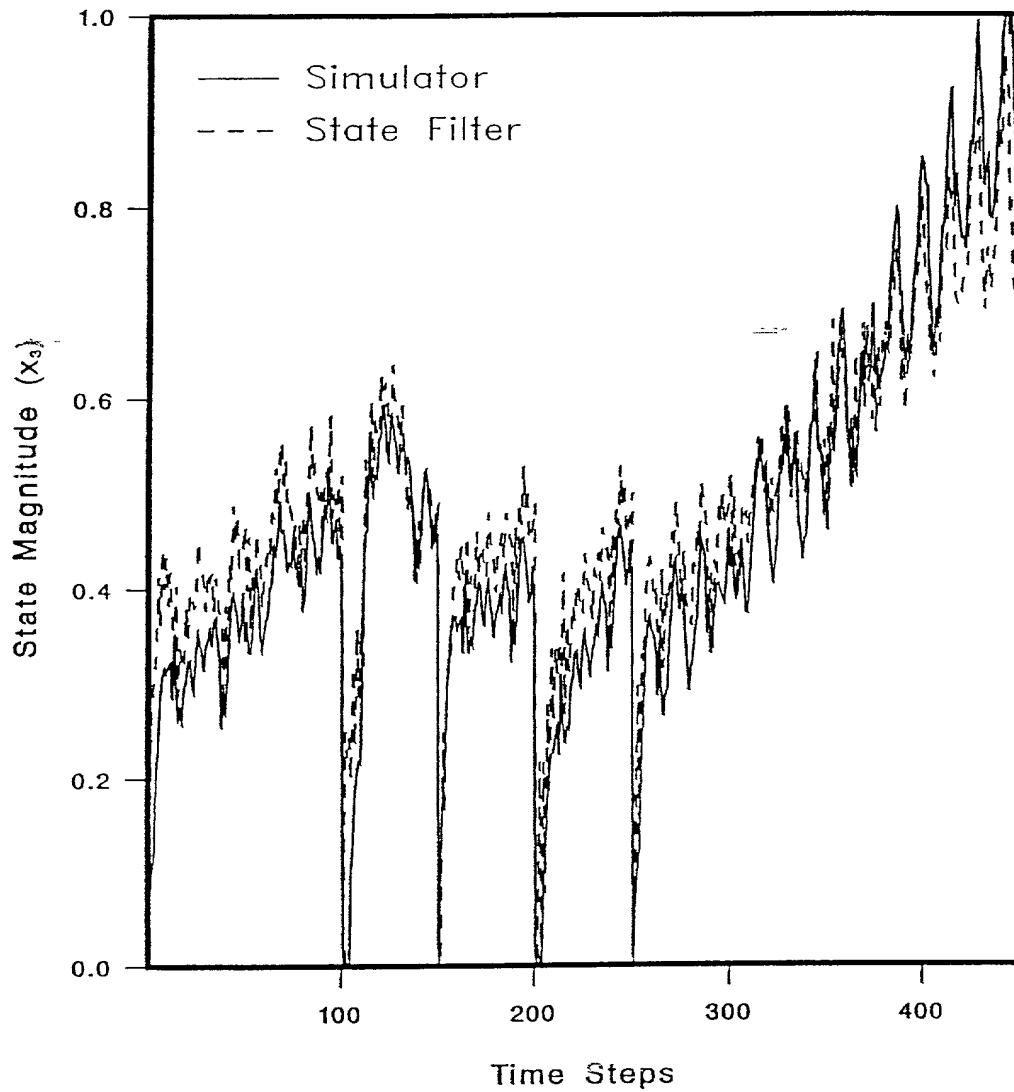


Figure 40. 2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter, for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).

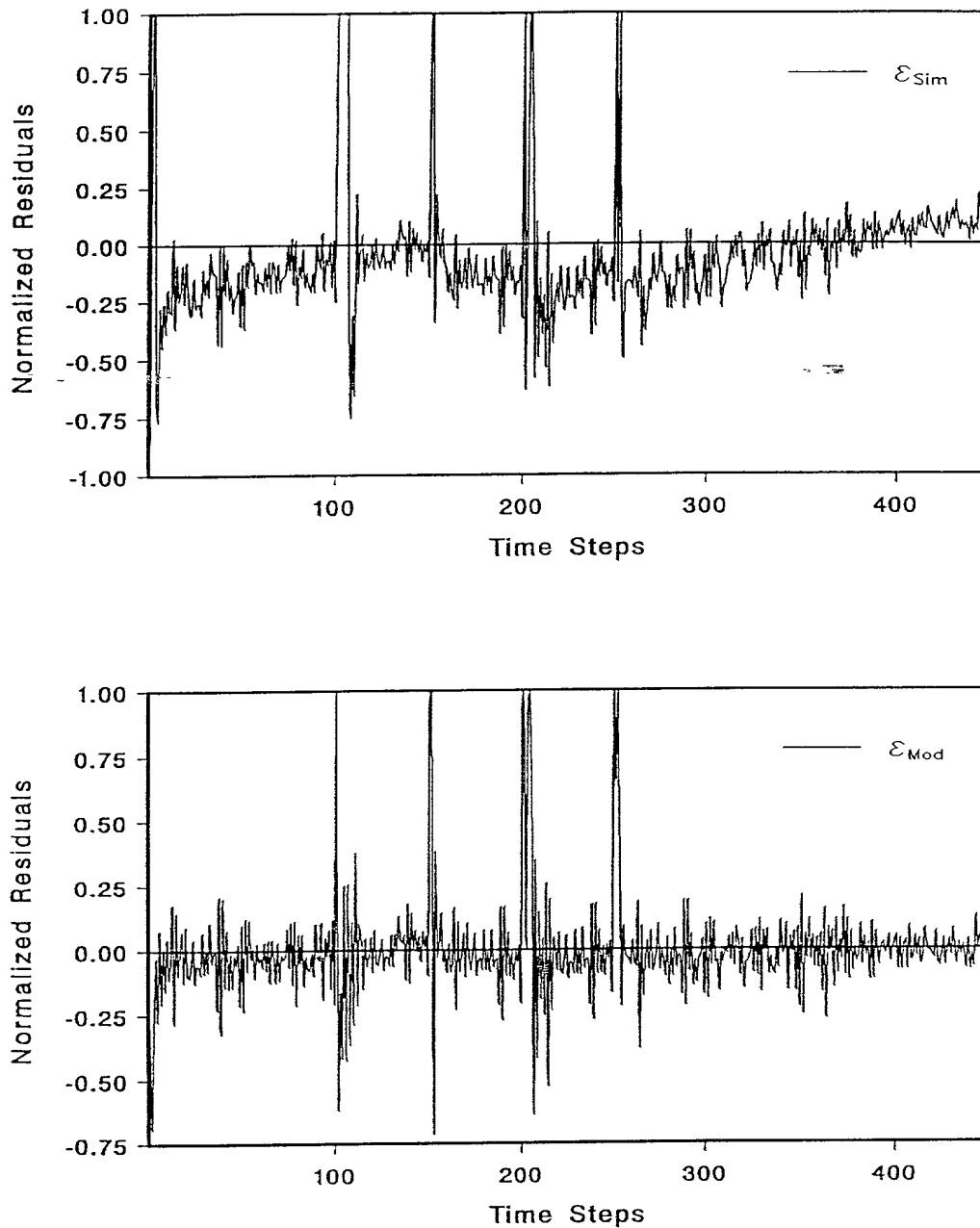


Figure 41. 2I2O State Filter Normalized Residuals Values for the Nonadaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).

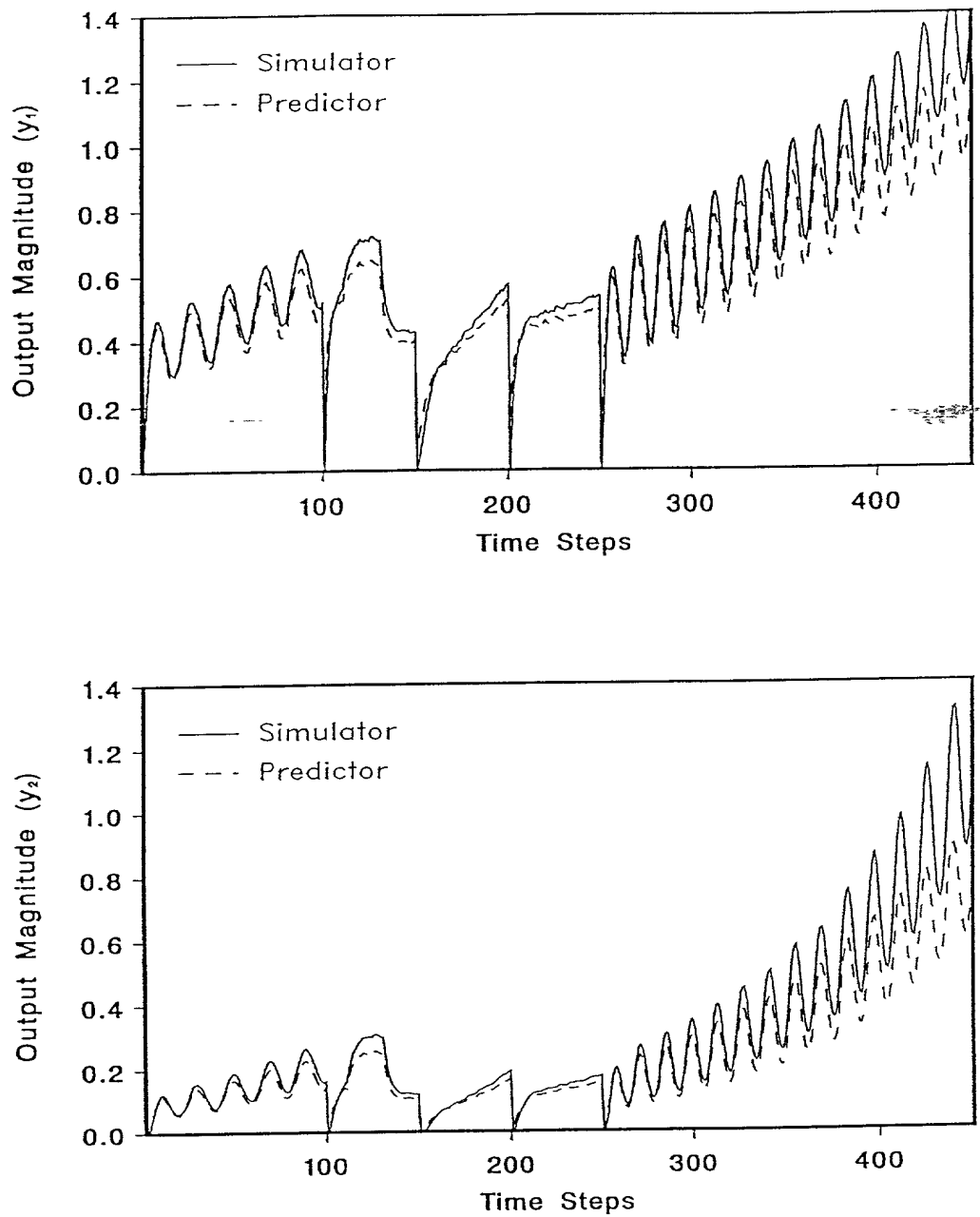


Figure 42. 2I2O System Output Response, from the Simulator and the NN Output Predictor, for the Adaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).

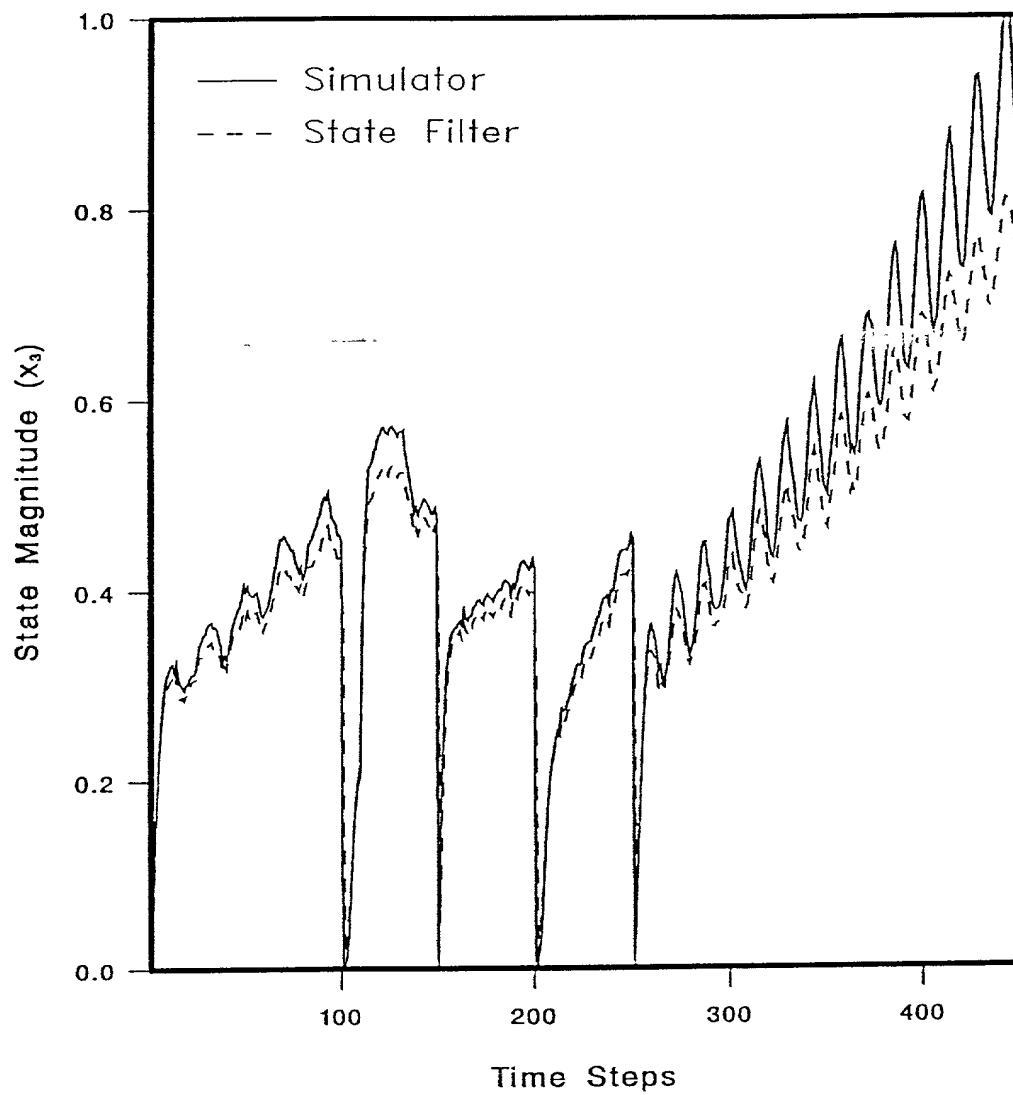


Figure 43. 2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter for the Adaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).

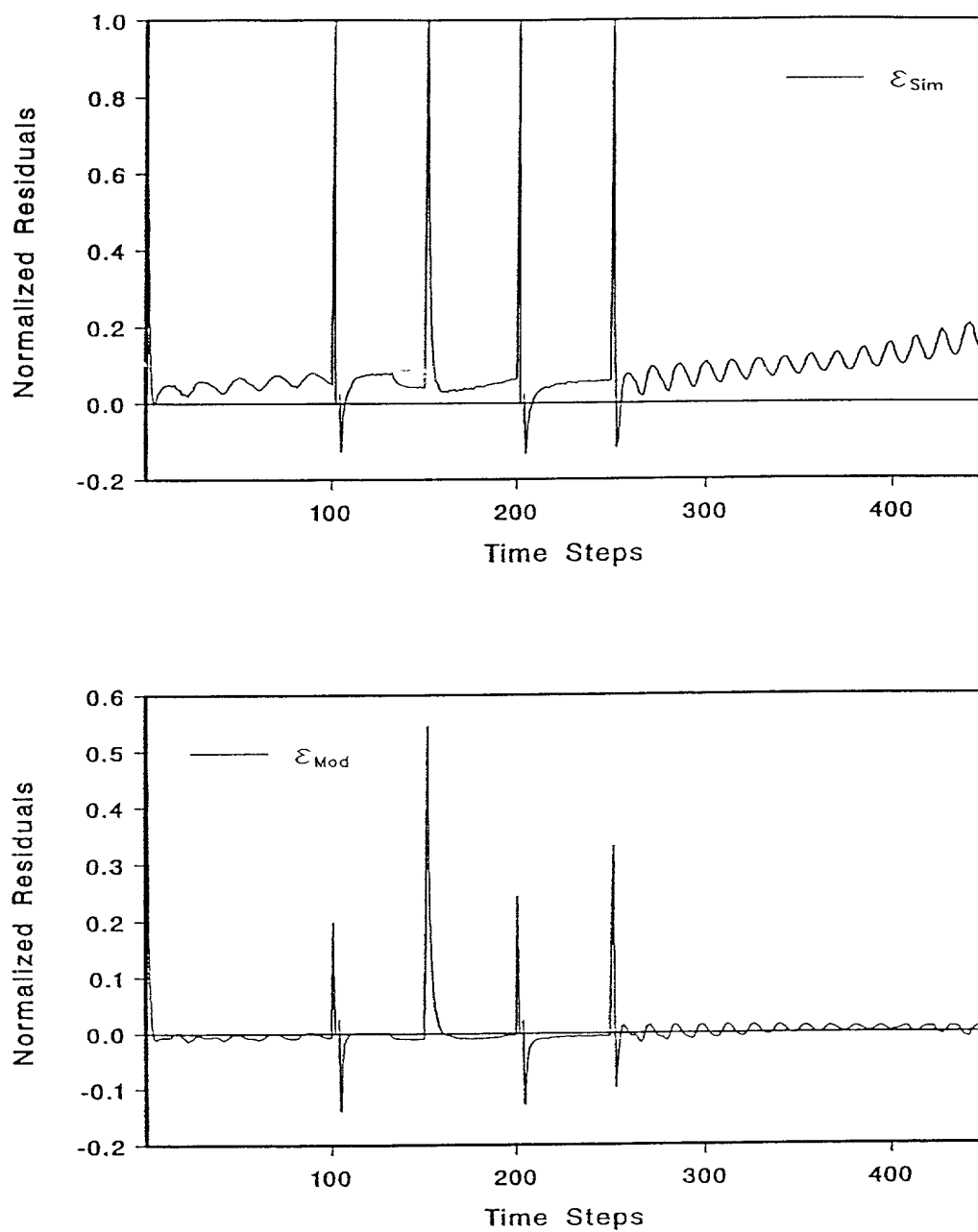


Figure 44. 2I2O State Filter Normalized Residuals for the Adaptive NN State Filter Using the Validation Data Set as Inputs (Low Noise Environment).

with sd 0.05). The output transient response of the NN output predictor is shown in Figure 45. The average NMSE of the filtered state values, $\hat{x}_3(t|t)$, is 0.26%. The filtered state values for the validation data set are shown in Figure 46, whereas the residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 47. It is seen that the effect of the higher level of noise in the validation data set did not adversely affect the accuracy of the state estimate $\hat{x}_3(t|t)$. This is possibly an indication of the noise rejection property of the NN. Nevertheless, the discrepancy between the residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, once again is indicative of the importance of model fidelity in the filter design.

V.6 Chapter Summary

A 2I2O system was chosen to demonstrate the nonadaptive and adaptive NN-based state filters developed in Chapter IV. The 2I2O system exhibits sufficiently complex nonlinearities, despite having a simple structure, to make this case study nontrivial. The 2I2O system (simulator) has three states, $x_1(t)$, $x_2(t)$, and $x_3(t)$, two inputs, $u_1(t)$ and $u_2(t)$, and two outputs, $y_1(t)$ and $y_2(t)$. The state to be estimated (filtered) is $x_3(t)$. Further, two system models, Model 1 and Model 2, are used in the nonadaptive and adaptive methods, respectively. In the nonadaptive filter design, Model 1 is used as a predictor and, further, it is used to generate state $x_{m3}(t)$, inputs and output values with which the NN state filter is trained. Model 2, is used in the the adaptive filter design, it is a more accurate than Model 1, and it is used, principally, to generate $x_{m3}(t)$, input and output values which are used to train the NN state predictors and state filter. In both methods, nonadaptive and adaptive, an estimation data set is used to train the NNs present in the filter design.

The nonadaptive and adaptive NN state filters are finally evaluated using a validation data set comprised of test signals different than those contained in the estimation data set. Further, the validation data set is corrupted with low and high values of process noise. The evaluation results are summarized in Table 3. The results from both state filters are good under both low and high noise levels. However, it should be noted that the performance of both the state filters is strongly dependent

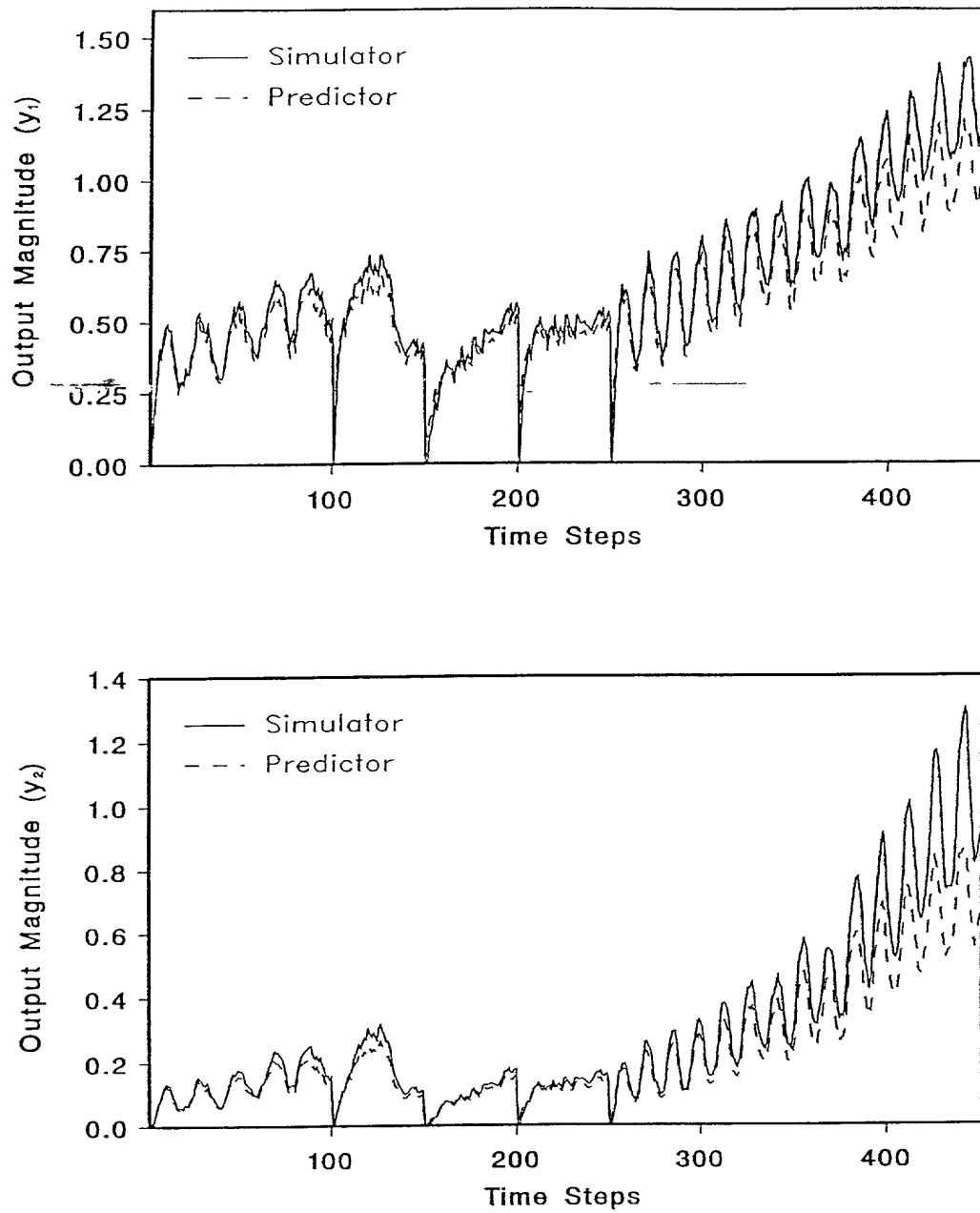


Figure 45. 2I2O System Output Response, from the Simulator and the NN Output Predictor, for the Adaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).

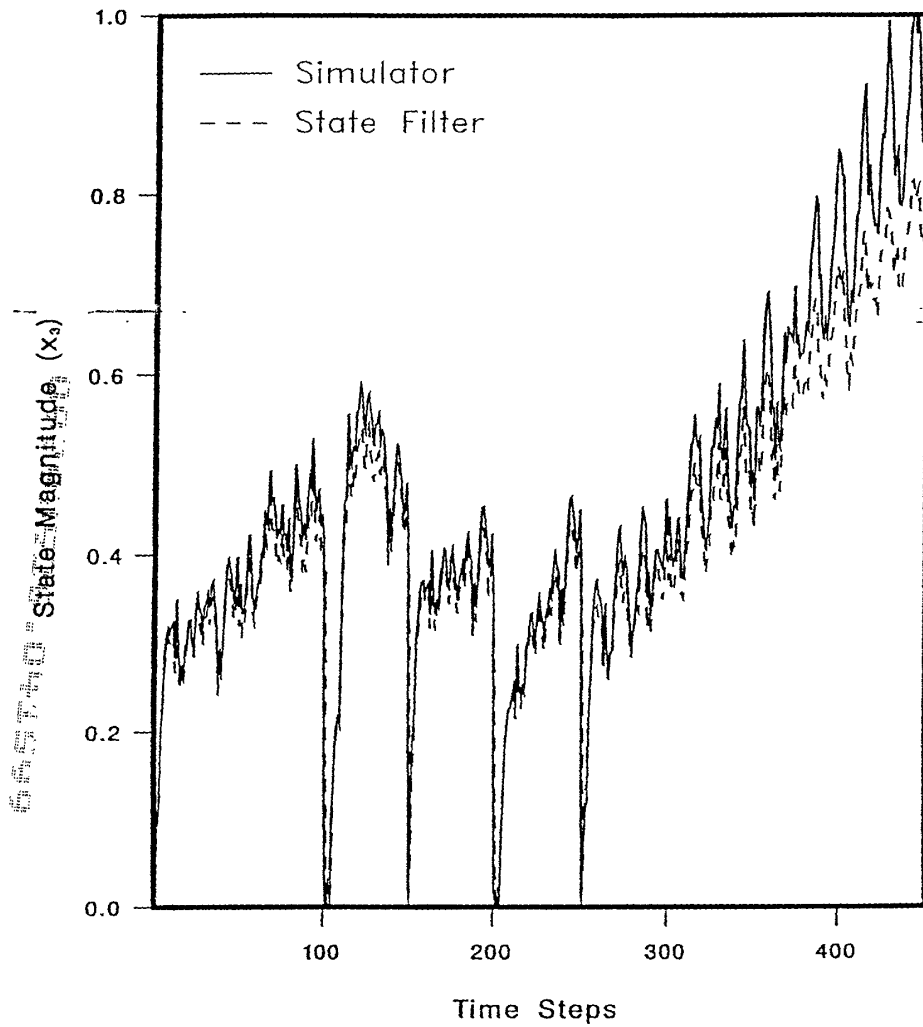


Figure 46. 2I2O System State $x_3(t)$ Response, from the Simulator and the NN State Filter, for the Adaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).

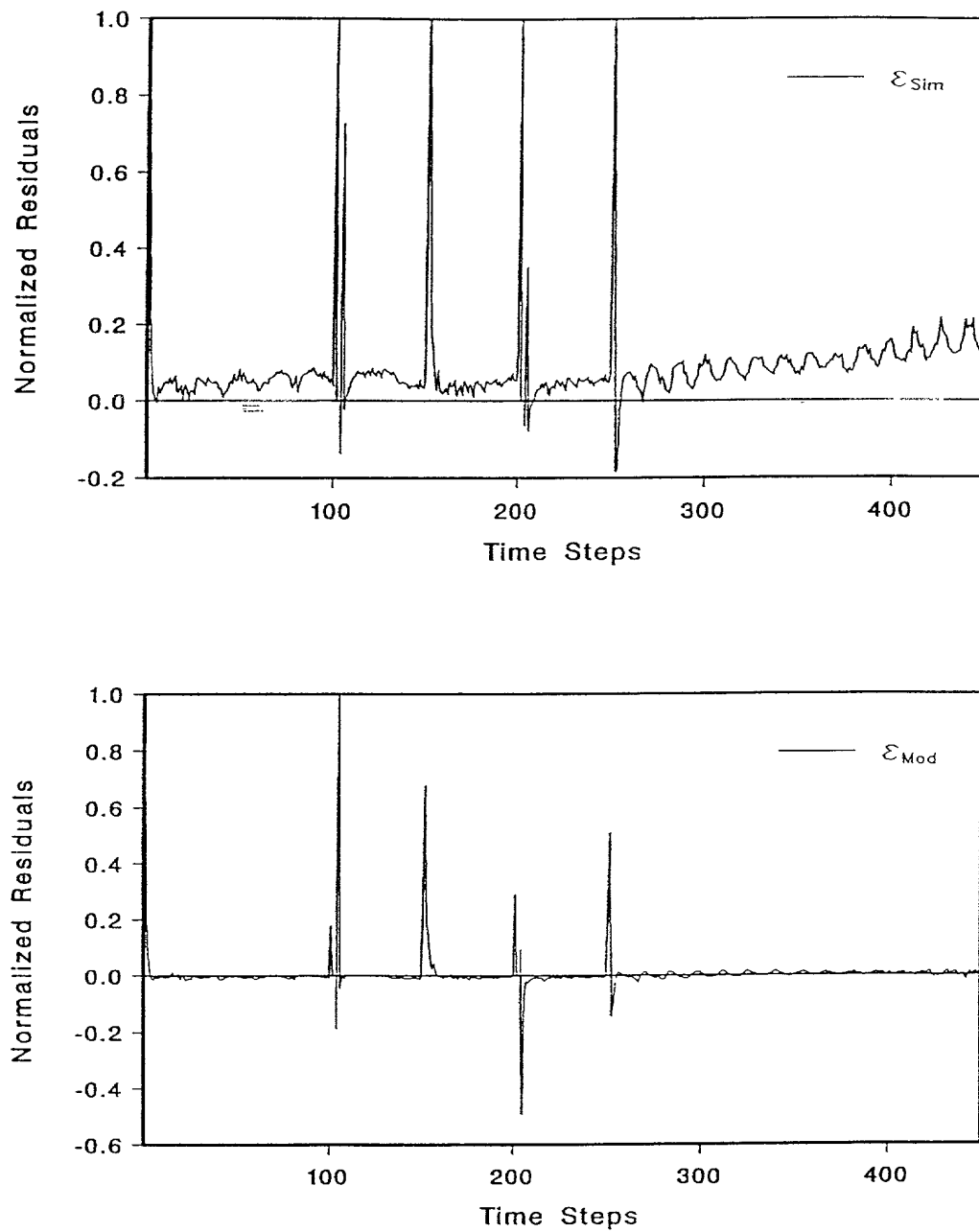


Figure 47. 2I2O State Filter Normalized Residuals for the Adaptive NN State Filter Using the Validation Data Set as Inputs (High Noise Environment).

on the fidelity of the system model used to generate the state, input and output values that are later used to train the NNs. In other words, the NN state filters are only as good as the quality of the state values used to train the NNs. Any further improvement in the filtered states must come from on-line adaptation using actual input and output measurements.

Table 3. 2I2O System Nonadaptive and Adaptive NN State Filter Results.

NN State Filter		$\overline{\text{NMSE}}$	$\overline{\epsilon_{\text{Sim}}(t)}$	$\overline{\epsilon_{\text{Mod}}(t)}$
Nonadaptive	Low Noise ($\sigma = 0.01$)	0.32%	-12.5%	-0.1%
	High Noise ($\sigma = 0.05$)	0.46%	-11%	-0.05%
Adaptive	Low Noise ($\sigma = 0.01$)	0.32%	10%	-0.1%
	High Noise ($\sigma = 0.05$)	0.26%	11%	-0.05%

APPLICATION 2 - DC MOTOR-PUMP SYSTEM

VI.1 Introduction

In this chapter, a DC Motor-Centrifugal Pump (Motor-Pump) system is used to develop the hybrid adaptive and ~~an~~ adaptive NN state filter, based on methods discussed in Chapter IV. The Motor-Pump is a relatively simple system, yet it exhibits sufficiently complex dynamics to make the development of the NN state filters realistic. In this study, the Motor-Pump system is represented by a simulator that was developed from first-principles.

The objective of this case study is to develop two NN state filters (hybrid adaptive and adaptive) for the Motor-Pump armature resistance, and an additional NN state filter for the simultaneous estimation of the Motor-Pump armature resistance and flux linkage.

This chapter is organized into 5 sections: The next section, VI.2, describes the Motor-Pump process simulator. Section VI.3 describes the first-principles model of the Motor-Pump process system, used to develop the NN state filters. Section VI.4 describes the development of the hybrid adaptive and adaptive NN state filters. Section VI.5 presents the results of the validation tests performed on the developed NN state filters. The final section, VI.6, summarizes the chapter.

VI.2 Process Description

The Motor-Pump process system comprises a DC motor, a centrifugal pump, and the associated piping system. A schematic diagram of the system is shown in

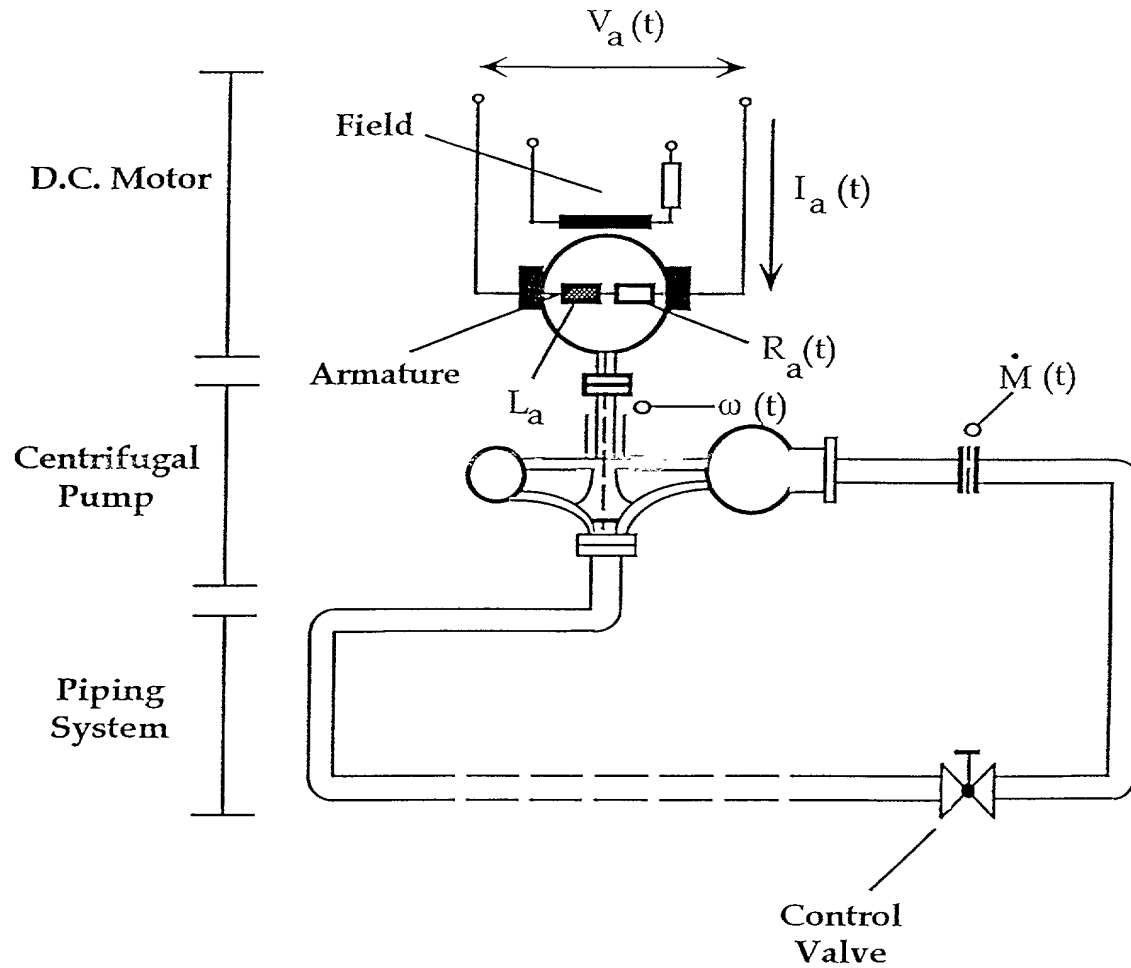


Figure 48. Schematic Diagram of the DC Motor, Centrifugal Pump and Associated Piping System.

Figure 48. A simulator is developed from first-principles to represent each of the components of the Motor-Pump system. The following subsections describe each of the lumped parameter models of the various process system simulator components [26],[27],[36].

VI.2.1 DC Motor

The DC motor speed is regulated by varying the armature current while the field current is kept constant. This study considers only the open-loop operation of the motor. Since the field current is excited separately, the magnetic flux is independent of the armature current and remains constant. The differential equation for the armature circuit is given by:

$$L_a \frac{dI_a(\tau)}{d\tau} + R_a(\tau) I_a(\tau) + V_b(\tau) = V_a, \quad (257)$$

where $R_a(\tau)$ is the armature resistance (Ω), L_a is the armature inductance (H), $I_a(\tau)$ is the armature current (A), and V_a is the armature voltage (V) which, in this study, is kept constant and $V_b(\tau)$ is the back emf (V). The armature resistance is assumed to vary with the armature current, or equivalently the external load demand, according to the following equation:

$$R_a(\tau) = \alpha \frac{dI_a^2(\tau)}{d\tau} + R_a^o, \quad (258)$$

where α is a constant and R_a^o is the nominal value of the armature resistance. Since the field current is constant, a constant flux linkage, Ψ , is generated, resulting in the following induced voltage $V_b(\tau)$:

$$V_b(\tau) = K_v \Psi \omega(\tau), \quad (259)$$

where $\omega(\tau)$ is the angular velocity of the motor shaft (rads/s), and K_v is a motor constant. For a constant field current, the torque of the motor becomes directly proportional to the armature current, as follows:

$$T_m(\tau) = K_t \Psi I_a(\tau), \quad (260)$$

where $T_m(\tau)$ is the mechanical torque developed by the motor (N-m), and K_t is the motor constant. The angular momentum conservation equation for the motor then becomes:

$$\theta_m \frac{d\omega(\tau)}{d\tau} = T_m(\tau) - T_{fm}(\tau) - T_s(\tau), \quad (261)$$

where θ_m is the moment of inertia of the motor ($kg - m^2$), $T_{fm}(\tau)$ is the friction torque of the motor (N-m), and $T_s(\tau)$ is the torque due to the load, (N-m). The friction torque can be attributed to the static and dynamic friction, while neglecting higher order terms, as follows:

$$T_{fm}(\tau) = c_{fm0} + c_{fm1}\omega(\tau), \quad (262)$$

where c_{fm0} and c_{fm1} are the static and dynamic friction coefficients of the motor, respectively.

VI.2.2 Centrifugal Pump

The angular momentum conservation equation for the centrifugal pump can be expressed as:

$$\theta_p \frac{d\omega(\tau)}{d\tau} = T_s(\tau) - T_p(\tau) - T_{fp}(\tau), \quad (263)$$

where θ_p is the moment of inertia of the pump ($kg - m^2$), $T_p(\tau)$ is the torque generated by the pump (N-m), and $T_{fp}(\tau)$ is the friction torque of the pump (N-m). The friction torque for the pump is assumed to be given by:

$$T_{fp}(\tau) = c_{fp0} + c_{fp1}\omega(\tau), \quad (264)$$

where c_{fp0} and c_{fp1} are the static and dynamic friction coefficients of the pump, respectively. Furthermore, the expression for $T_p(\tau)$ is given by:

$$T_p(\tau) = h_2 \dot{M}(\tau) \omega(\tau), \quad (265)$$

where h_2 is the pump constant, and $\dot{M}(\tau)$ is the mass flow rate (kg/s).

Eliminating $T_s(t)$ from equations (261) and (263), results in the combined conservation equation for the angular momentum of the DC motor and the centrifugal pump, as follows:

$$\theta_{mp} \frac{d\omega(\tau)}{d\tau} = \Psi I_a(\tau) - c_{mp0} - c_{mp1}\omega(\tau) - h_2 \dot{M}(\tau) \omega(\tau), \quad (266)$$

where, $\theta_{mp} = \theta_m + \theta_p$, $c_{mp0} = c_{fm0} + c_{fp0}$, and $c_{mp1} = c_{fm1} + c_{fp1}$.

VI.2.3 Piping System

The piping system consists of a pipe of length L , a valve, and the fluid resistances in the pipe. The steady-state conservation equation for the momentum of the piping system can be expressed as follows:

$$\Delta p_p(\tau) - \Delta p_r(\tau) - \Delta p_v(\tau) + \Delta p_{ac}(\tau) = 0, \quad (267)$$

where $\Delta p_p(\tau)$ is the pressure rise across the pump (Pa), $\Delta p_r(\tau)$ is the pressure drop due to fluid resistances (lumped), (Pa), $\Delta p_v(\tau)$ is the pressure drop due to the valve, (Pa), and, $\Delta p_{ac}(\tau)$ is the pressure drop due to the acceleration of the fluid, (Pa). The pressure drops due to the fluid resistances, the control valve, and the acceleration of the fluid through the pipe can be expressed as follows:

$$\Delta p_r(\tau) = c_r \dot{M}^2(\tau), \quad (268)$$

$$\Delta p_v(\tau) = c_v \left(\frac{\dot{M}(\tau)}{A_v} \right)^2, \quad (269)$$

$$\Delta p_{ac}(\tau) = -\rho L \frac{dv(\tau)}{d\tau} = -\frac{L}{A_p} \frac{d\dot{M}(\tau)}{dt}, \quad (270)$$

respectively, where, A_p is the cross-sectional area of the pipe, (m^2), A_v is the cross-sectional area of the valve, (m^2), $v(\tau)$ is the fluid velocity, (m/s), L is the pipe length, (m), c_v is the valve coefficient, and c_r is the fluid resistance coefficient. Substituting equations (268), (269), and (270), in equation (267) results in the following conservation equation for the piping system:

$$\frac{L}{A_p \rho} \frac{d\dot{M}(\tau)}{d\tau} + \left(\frac{c_v}{A_v^2 \rho} + \frac{c_r}{\rho} \right) \dot{M}^2(\tau) = Y(\tau), \quad (271)$$

where $Y(\tau) = (\Delta p_p(\tau)/\rho)$ is the specific energy, (m^2/s^2). Furthermore, $Y(\tau)$ can be expressed as:

$$Y(\tau) = h_{nn} \frac{A_p \rho}{L} \dot{M}^2(\tau), \quad (272)$$

where h_{nn} is a pump characteristic parameter. Finally, equation (271) can be expressed as:

$$\frac{d\dot{M}(\tau)}{d\tau} = h_{nn} \dot{M}^2(\tau) - h_{rr}(\tau) \dot{M}^2(\tau), \quad (273)$$

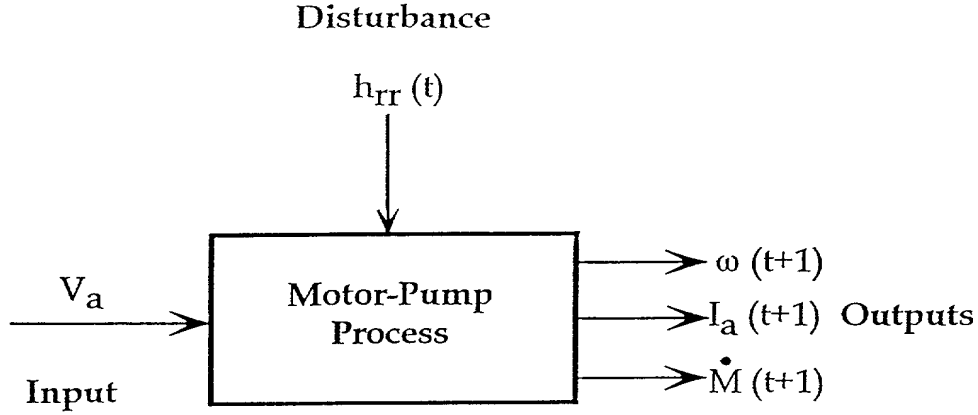


Figure 49. Block Diagram of the Motor-Pump Process System.

where $h_{rr}(\tau)$ is a time-varying lumped parameter that models the flow resistance due to the valve and pipe circuit. In the current study, the external load on the overall system is varied through variations in $h_{rr}(\tau)$. The external load cannot be directly measured, however, and it must be estimated. This is described in further detail in the following subsections.

Three of the states of the Motor-Pump system, $\omega(t)$, $I_a(t)$ and $\dot{M}(t)$, are measured. The armature resistance, $R_a(t)$, cannot be measured directly and it must be estimated. The numerical values of the constant coefficients used in the above equations are given as follows: $L_a = 0.08\text{H}$, $V_a = 60.3\text{V}$, $K_v = 1$, $\theta_{mp} = 1.16$, $\Psi = 5.8\text{Wb}$, $c_{mp0} = 2.2$, $c_{mp1} = 1.16$, $h_2 = 0.42$, $h_{nn} = 0.019$, and $\alpha = 0.0625$. The steady-state values of the states are: $I_a^0 = 4.04\text{A}$, $\omega^0 = 9.71\text{rads/s}$, $\dot{M} = 2.448\text{kg/s}$, and $R_a^0 = 1.45\Omega$. A block diagram of the Motor-Pump system in discrete-time form is shown in Figure 49.

VI.3 Process Model Description

The Motor-Pump process model can be represented by the following coupled differential equations, obtained by rewriting equations (257), (259), (266) and (273) (the

subscript m referring to the model):

$$L_{ma} \frac{dI_{ma}(\tau)}{d\tau} = V_{ma} - R_{ma}(\tau) I_{ma}(\tau) - K_{mv} \Psi_m \omega_m(\tau), \quad (274)$$

$$\theta_{mmp} \frac{d\omega_m(\tau)}{d\tau} = \Psi_m I_{ma}(\tau) - c_{mmp0} - c_{mmp1} \omega_m(\tau) - \hat{T}_L(\tau), \quad (275)$$

$$\dot{M}_m(\tau) = K_{mM} \omega_m(\tau), \quad (276)$$

$$R_{ma}(\tau) = \alpha_m \frac{dI_{ma}^2(\tau)}{d\tau} + R_{ma}^o, \quad (277)$$

where K_M is an empirically obtained coefficient and the other constant coefficients have the same significance as in the process system equations. The numerical values of the constant coefficients used in the above equations are given as follows: $L_{ma} = 0.07H$, $V_{ma} = 60.3V$, $K_{mv} = 0.9$, $\theta_{mmp} = 1.1$, $\Psi = 5.8Wb$, $c_{mmp0} = 2$, $c_{mmp1} = 1.1$, $K_{mM} = 0.25$, and $\alpha_m = 0.0625$. The steady-state values of the states are: $\hat{I}_a^0 = 4.04A$, $\hat{\omega}^0 = 9.71 \text{ rads/s}$, $\hat{M} = 2.448 \text{ kg/s}$, and $\hat{R}_a^0 = 1.3\Omega$.

The difference in the constant coefficients between the simulator equations and the process model equations accounts for some of the mismatch in the process model output and the process system output. In addition, the process model equations use an empirical relationship, expressed by equation (276), to model the piping system. The load (external disturbance) on the system, represented by $\hat{T}_L(t)$ is estimated from the measured signals, $\omega(t)$ and $\dot{M}(t)$, using a semi-empirical relationship represented as follows:

$$\hat{T}_L(t) = k \dot{M}(t) \omega(t), \quad (278)$$

where k is a constant that is empirically determined. The value of k that is used is 0.36. The estimated load, $\hat{T}_L(t)$, is also referred to as $\hat{u}(t)$ and is one of the inputs used in the NN state filters.

The equations describing the *predictor* form of the process model are represented as follows:

$$\hat{I}_a(t+1|t) = f_{mp1}(\hat{I}_a(t|t), \hat{\omega}(t|t), \hat{R}_a(t|t)), \quad (279)$$

$$\hat{\omega}(t+1|t) = f_{mp2}(\hat{I}_a(t|t), \hat{\omega}(t|t), \hat{T}_L(t)), \quad (280)$$

$$\hat{M}(t+1|t) = f_{mp3}(\hat{\omega}(t|t)), \quad (281)$$

$$\hat{R}_a(t+1|t) = f_{mp4}(\hat{I}_a(t|t)), \quad (282)$$

where the functions $f_{mp1}(\cdot)$, $f_{mp2}(\cdot)$, $f_{mp3}(\cdot)$, and $f_{mp4}(\cdot)$ are the discrete-time forms of the functions used in the process model differential equations represented by equations (274), (275), (276), and (277).

VI.4 Neural Network State Filters

In this section, hybrid adaptive and adaptive NN state filters are developed for the Motor-Pump system. All of the state filters use the same process model as described by equations (279), (280), (281), and (282).

Hybrid adaptive and adaptive NN state filters, based on the methods developed in Chapter IV, are designed to estimate the armature resistance, $R_a(t)$. The estimation and validation data sets used in the design and validation of the NNs involved are a collection of sinusoids and ramps. The sinusoids are of varying amplitudes and frequencies, and the ramps are of varying ramp rates, both positive and negative. In addition, process noise is added to the equations while data is collected. The process noise is zero-mean, white, Gaussian with 0.02 sd. The estimation and the validation data sets used in the hybrid adaptive NN state filter are presented in Table 4. The data samples are collected every 0.2 seconds ($\Delta t = 0.2s$). The total number of samples in the estimation data set is 5000. Of these 5000 samples, the NN weights are updated using only the first 2500 samples, while the remaining 2500 samples are used to evaluate the generalization of the NN models. The estimation data set used in the adaptive state filter is the same as that used in the hybrid adaptive state filter with the exception that the ninth transient ($0.3 + 0.1 \sin(t/1.2)$) in Table 4 is omitted. The validation data set used in the adaptive state filter, however, is exactly the same as the validation data set used in the hybrid adaptive filter. Therefore, for the adaptive state filter, the total number of samples in the training data set is 4500. Out of these 4500 samples, the NN weights are updated using only the first 2500 samples, and the remaining 2000 samples are used to evaluate the generalization of the NN models. As seen in Table 4, the total number of samples in the validation data set used for both hybrid adaptive and adaptive filters is 1000.

Table 4. Motor-Pump System Estimation and Validation Data Set Used by the Hybrid Adaptive and Adaptive NN Armature Resistance Filters.

Estimation Data Set		
Weight Update	Load Variation, $h_{rr}(t)$	Number of Samples ($\Delta t = 0.2s$)
Yes	$0.3 + 0.2 \sin(t)$	500
	$0.3 + 0.2 \sin(t/3)$	500
	$0.3 + 0.25 \sin(t/2)$	500
	$0.3 + 0.2(t/40), \quad t \leq 40s$ $0.5, \quad t \in [40s, 50s]$ $0.5 - 0.15(t/40), \quad t \in [51s, 100s]$	500
	$0.3 + 0.1 \sin(2t)$	500
No	$0.3 + 0.2 \sin(t/1.5)$	500
	$0.3 + 0.1 \sin(t/2.5)$	500
	$0.3 + 0.1 \sin(10t/7)$	500
	$0.3 + 0.1 \sin(t/1.2)$	500
	$0.3 + 0.2(t/60), \quad t < 40s$ $0.41, \quad t \in [40s, 50s]$ $0.41 - 0.2(t/60), \quad t \in [51s, 100s]$	500
Validation Data Set		
No	$0.3 + 0.25 \sin(t/12)$	500
	$0.3 + 0.25(t/100), \quad t < 30s$ $0.45, \quad t \in [30s, 40s]$ $0.45 - 0.2(t/120), \quad t \in [40s, 100s]$	500

A hybrid adaptive state filter is also designed to estimate *both* the armature resistance, $R_a(t)$, as well as the constant flux linkage, Ψ , and is discussed separately. The estimation and validation data sets for this two output filter are also a collection of sinusoids and ramps as for the previous filters. However, in this case the process model used in this case is different from the process model used in the previous filters in that it is more accurate (closer to the process simulator). This was done in order to evaluate the merits of the two output hybrid adaptive NN state filter without the filtering complications caused by modeling inaccuracies. The process model constant

Table 5. Motor-Pump System Estimation and Validation Data Set Used by the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter.

Estimation Data Set		
Weight Update	Flux Linkage Ψ (Wb)	Load Variation, $h_{rr}(t)$
Yes	5.8,6.5,5.1	$0.3 + 0.2 \sin(t)$
		$0.3 + 0.2 \sin(t/3)$
		$0.3 + 0.25 \sin(t/2)$
		$0.3 + 0.2 (t/40), \quad t \leq 40s$
		$0.5, \quad t \in [40s, 50s]$
		$0.5 - 0.15 (t/40), \quad t \in [51s, 100s]$
No	5.8	$0.3 + 0.1 \sin(2t)$
		$0.3 + 0.2 \sin(t/1.5)$
	6.1	$0.3 + 0.1 \sin(t/2.5)$
		$0.3 + 0.1 \sin(10t/7)$
	5.4	$0.3 + 0.1 \sin(t/1.2)$
		$0.3 + 0.2 (t/60), \quad t < 40s$ $0.41, \quad t \in [40s, 50s]$ $0.41 - 0.2 (t/60), \quad t \in [51s, 100s]$
Validation Data Set		
No	5.9	$0.3 + 0.25 \sin(t/12)$
	5.5	$0.3 + 0.25 (t/100), \quad t < 30s$ $0.45, \quad t \in [30s, 40s]$ $0.45 - 0.2 (t/120), \quad t \in [40s, 100s]$

coefficients used are not reported here; however, they can be assumed to be the same as that of constant coefficients of the simulator.

In the design of the two output hybrid adaptive state filter, the data sets are collected for *different* constant values of Ψ . As before, process noise is added to the equations while data is collected. The process noise is zero-mean, white, Gaussian with 0.02 sd. The estimation and validation data sets are listed in Table 5. In this hybrid adaptive filter, the total number of samples in the estimation data set is 10000. Out of these 10000 samples, the NN weights are updated using only the first 7500 samples, and the remaining 2500 samples are used to evaluate the generalization of

the NN models. As seen in Table 5, the number of samples used in the validation data set of this hybrid adaptive filter development is 1000.

The following subsections discuss in detail, the design and evaluation of both hybrid adaptive filters and the adaptive state filters, as applied to the Motor-Pump system. All simulations of the NNs were performed on an SGI Power Challenge XLTM computer.

VI.4.1 Hybrid Adaptive Filter for the Armature Resistance

The method used to develop the armature resistance hybrid adaptive state filter is described in Chapter IV. The vector $y(t)$ will be used to denote the three outputs of the Motor-Pump process system simulator: $\omega(t)$, $I_a(t)$, and $\dot{M}(t)$. The vector $\hat{y}(t)$ refers to the three outputs of the process model (predictor): $\hat{\omega}(t)$, $\hat{I}_a(t)$, and $\hat{\dot{M}}(t)$. The input to the predictor is $\hat{u}(t)$ which is the estimate of the load disturbance, $T_L(t)$, whereas the armature voltage, V_a , is assumed constant. The NN error model, and the NN state filter development is discussed in the following subsections.

VI.4.1.1 NN Error Model Development

The NN error model is first chosen to be a three-layer FMLP network with 7 nodes in the first layer, corresponding to the 7 inputs: $\hat{u}(t)$, $y(t+1)$ and $\hat{y}_e(t|t)$. The vector $\hat{y}_e(t|t)$ is the difference between the process measurements, $y(t)$, and the model predictor output, $\hat{y}(t|t-1)$. There are three nodes in the third (output) layer, corresponding to the three outputs $\hat{y}_e(t+1|t+1)$. The number of nodes in the second (hidden) layer is initially set to 3. The NN error model is then trained with the estimation data set. The target is $y_e(t+1)$, which is computed by $y(t+1) - \hat{y}(t+1|t)$.

Training is stopped when the NMSE of the estimation data set (no weight update part) just begins to increase (indicating the start of over-training). The NMSE is noted, the number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. The NN architecture with the lowest NMSE in the estimation data set (no weight update) is

chosen. In this case, this NN architecture was 7-4-3 with a NMSE of 0.182%. The RMLP network was not used to develop the NN error model because the low NMSE of the FMLP networks did not warrant the use of the more complex RMLP networks.

VI.4.1.2 NN State Filter Development

The NN state filter architecture is first chosen to be a three-layer FMLP network with 7 nodes in the first layer, corresponding to 7 inputs, and 1 node in the third layer, corresponding to 1 output. The 7 inputs to the first layer are the three process outputs, $y(t+1)$, the three residuals, $\epsilon(t+1)$ where $\epsilon(t+1) = y(t+1) - \hat{y}_c(t+1|t+1)$, and the state $R_{ma}(t)$. The vector $\hat{y}_c(t+1|t+1)$ is the *corrected* model predictor output, and it is the sum of the model predictor output, $\hat{y}(t+1|t)$, and the output of the NN error model, $\hat{y}_e(t+1|t+1)$. The single filter output is $\hat{R}_{NN,a}(t+1|t+1)$. The number of nodes in the hidden layer is initially chosen to be 3. The NN state filter is then trained with the estimation data set. The target is the state $R_{ma}(t+1)$ as provided by the Motor-Pump process model.

Training is stopped when the NMSE of the estimation data set (no weight update) just begins to increase (indicating the start of over-training). The NMSE is noted, the number of nodes in the hidden layer is increased, and the training cycle is repeated. In this case, it was determined that the FMLP NN with an architecture of 7-3-1 was the best network. The NMSE for the estimation data set (no weight update) for this network is 0.51%. A block diagram of the hybrid adaptive filter implementation is shown Figure 50. A comparison between the actual Motor-Pump process outputs, $y(t)$, and the corrected Model Predictor outputs, $\hat{y}_c(t|t)$, is shown in Figure 51. The comparison between the actual process state values, $R_a(t)$, the process model values, $R_{ma}(t)$, and the FMLP NN state filter output, $\hat{R}_a(t|t)$, is shown in Figure 52.

The RMLP network was not used to develop this NN state filter because the FMLP network performed adequately and the additional complexity of an RMLP network was neither justified nor warranted.

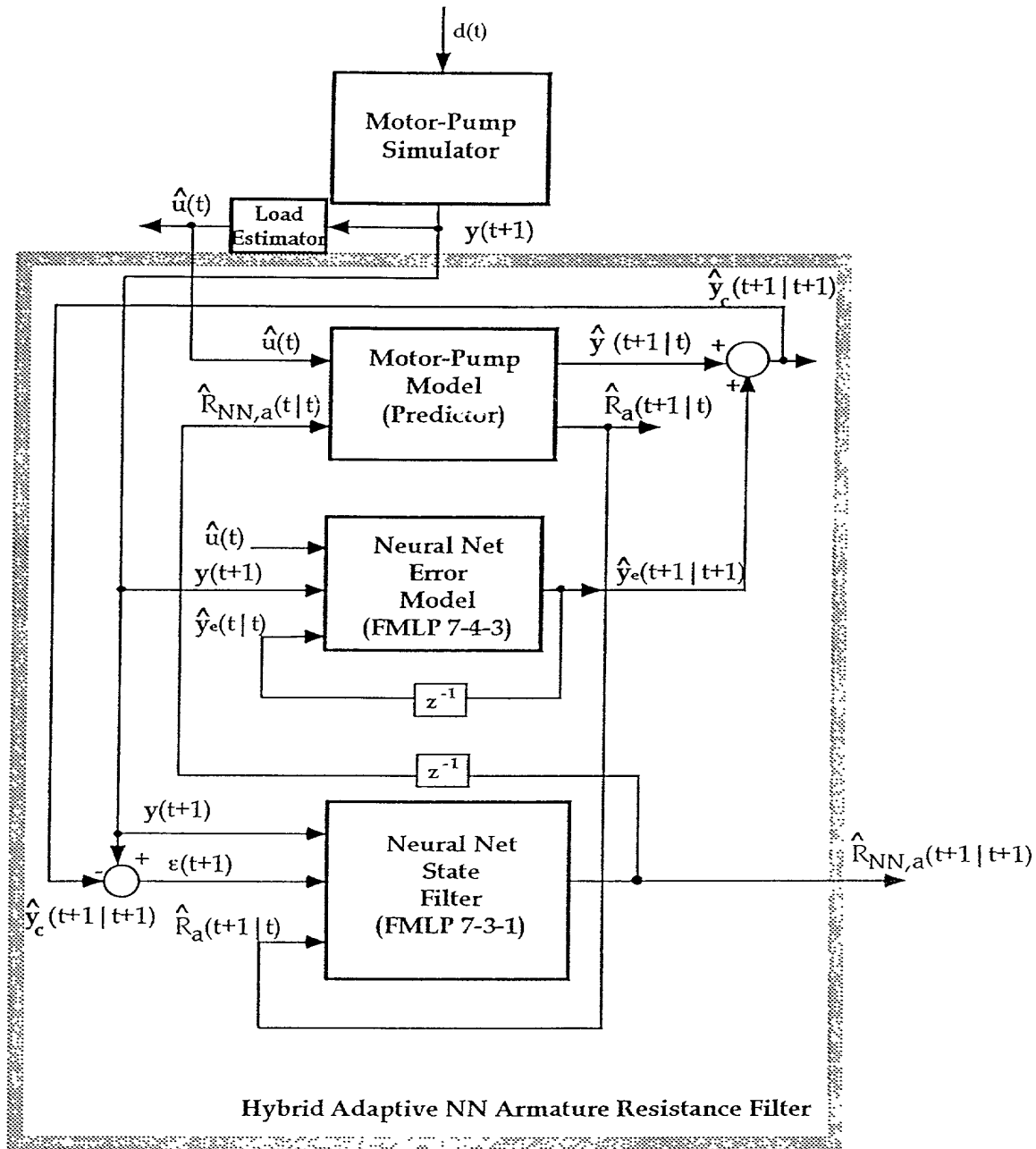


Figure 50. Block Diagram of the Motor-Pump System Hybrid Adaptive NN Armature Resistance Filter.

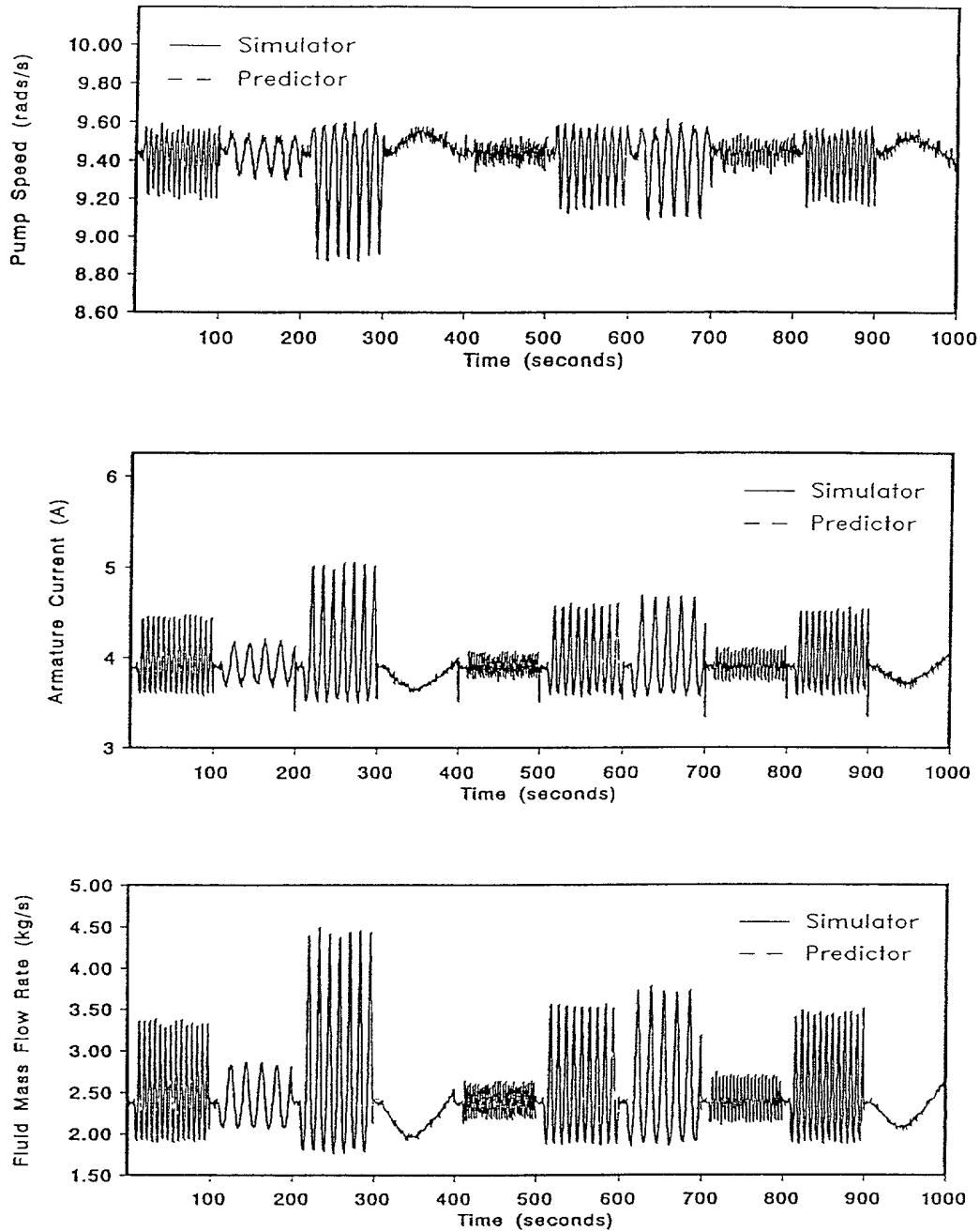


Figure 51. Motor-Pump System Output Response, from the Simulator and System Model, for the Hybrid Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.

495

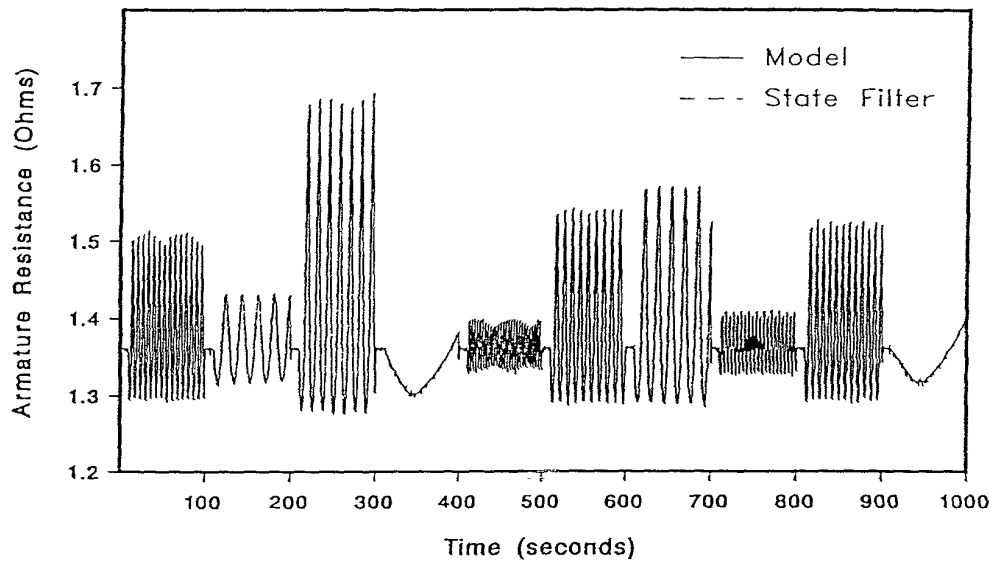
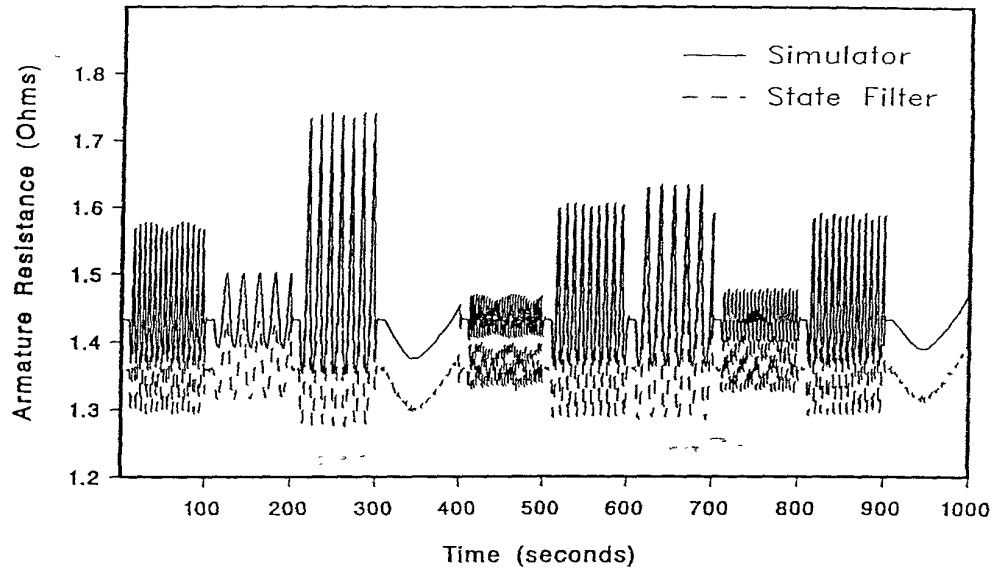


Figure 52. Motor-Pump System Armature Resistance Response, from the Simulator, Process Model, and the NN State Filter, for the Hybrid Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.

VI.4.2 Adaptive Filter for the Armature Resistance

The method used to develop the armature resistance adaptive filter is described in Chapter IV. Three NNs are used in this filtering method: the NN output predictor, the NN state predictor and the NN state filter. The NN output predictor is developed first. The NN state predictor is developed then, with the NN output predictor running concurrently. Finally, the NN state filter is developed with both the NN output predictor, and the NN state predictor running concurrently. The development of each of these NN is discussed in greater detail in the following subsections.

VI.4.2.1 NN Output Predictor Development

The NN output predictor comprises three NNs; one NN is developed for each of the outputs of the Motor-Pump process. Each of the NNs is a three layer RMLP network. The inputs for all three NNs are the same and they are 5 in number. These are: the estimate of the load applied on the process system, $\hat{u}(t)$; the filtered value of the armature resistance, $\hat{R}_{NN,a}(t|t)$; and the NN output predictor outputs (delayed by one time step), $\hat{y}_{NN}(t|t-1)$. The output of the NNs is $\hat{\omega}_{NN}(t+1|t)$, $\hat{I}_{NN,a}(t+1|t)$ and $\hat{M}_{NN}(t+1|t)$, respectively. The number of nodes in each of the NNs is set initially to 5. The three NN models are then trained in parallel with the estimation data set. The targets for each of the NNs, collected from the process model, are $\omega_m(t+1)$, $I_{ma}(t+1)$ and $\dot{M}_m(t+1)$, respectively. During training, the input $\hat{R}_{NN,a}(t|t)$ is replaced by the process model resistance, $R_{ma}(t)$. Training is stopped when the NMSE, on the estimation data set (with no weight update), of the network outputs just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A suitable number of NN architectures are obtained in this manner and the best among them are chosen. In this case, the best NN model architecture was found be 5-6-1 for each of the NNs in the NN output predictor. A block diagram of

the output predictor is shown in Figure 53. The NMSEs of the three NNs modeling the three process outputs are: 0.007% for the angular velocity ($\omega(t)$), 0.2% for the armature current ($I_a(t)$), and 1.2% for the mass flow rate ($\dot{M}(t)$). These NMSE values are evaluated on the estimation data set (no weight update). A comparison between the Motor-Pump process simulator outputs and the NN output predictor outputs for the estimation data set transients are shown in Figure 54.

VI.4.2.2 NN State Predictor Development

The NN output predictor weights are now fixed and no further training is performed on this model. The NN state predictor is initially chosen to be a three layer RMLP with 5 nodes in the first layer, corresponding to 5 inputs, and 1 node in the third layer, corresponding to one output. The output of the third layer is the SSP of the state, $\hat{R}_{NN,a}(t+1|t)$. The inputs to the first layer, which are the same as the inputs to the NN output predictor, are the delayed outputs from the NN output predictor, $\hat{y}(t|t-1)$, the estimate of the process system load, $\hat{u}(t)$, and the delayed output of the NN state predictor, $\hat{R}_{NN,a}(t|t-1)$. The number of nodes in the hidden layer is initially set to 3. The NN state predictor is then trained with the estimation data set. The target is the state $R_{ma}(t+1)$, as collected from the Motor-Pump process model.

The NN state predictor is trained until the NMSE of the estimation data set (no weight update) just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. Based on the lowest NMSE on the training data set (no weight update), a RMLP network with an architecture of 5-4-1 was determined to be the best. The average NMSE for this architecture was 7.5%. FMLP networks were not used to develop the NN state predictor because the complexity of this process system necessitates the use of the RMLP networks.

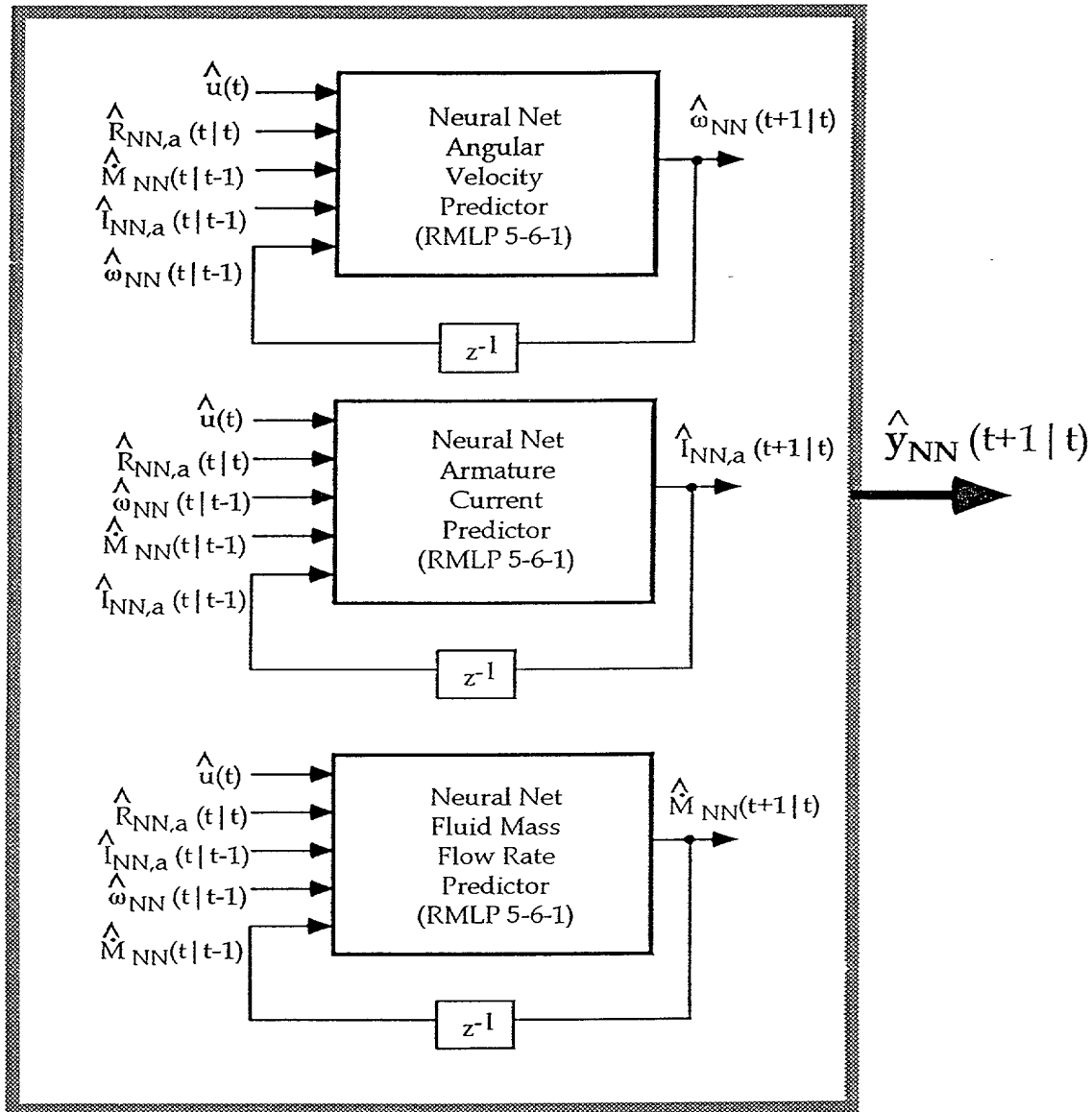


Figure 53. Block Diagram of the Motor-Pump System NN Output Predictor Used in the Adaptive NN Armature Resistance Filter.

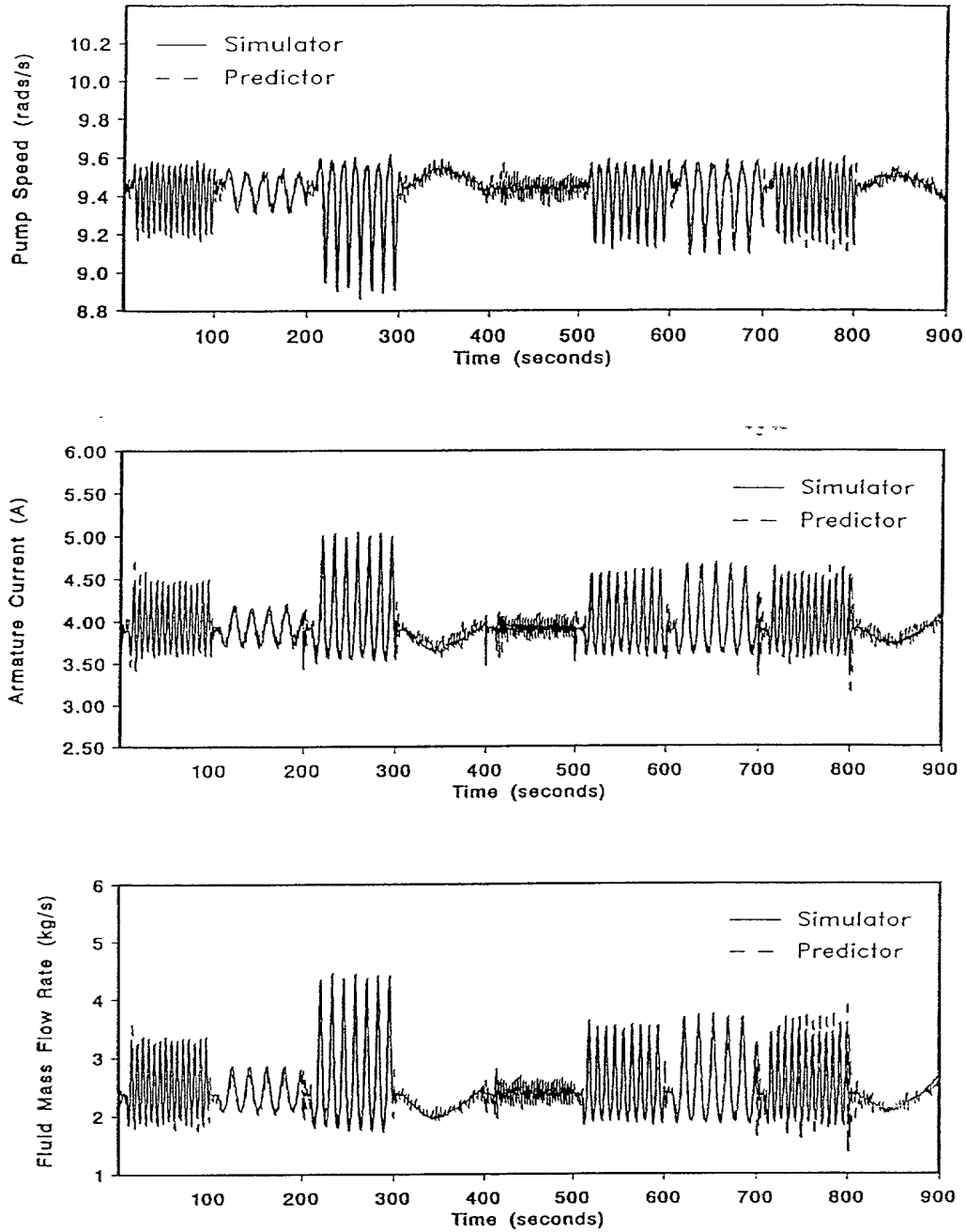


Figure 54. Motor-Pump System Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.

VI.4.2.3 NN State Filter Development

The weights of the NN output predictor and NN state predictor are now fixed and no further training is performed on these networks. The NN state filter architecture is first chosen to be a three-layer FMLP NN with 7 nodes in the input layer, corresponding to 7 inputs, and 1 node in the third layer, corresponding to one output. The output of the third layer is the SSP of the resistance, $\hat{R}_{NN,a}(t+1|t+1)$. The 7 inputs to the first layer are the Motor-Pump process outputs, $y(t+1)$, the residuals, $\epsilon(t+1)$ where $\epsilon(t+1) = y(t+1) - \hat{y}_{NN}(t+1|t)$, and the output from the NN state predictor, $\hat{R}_{NN,a}(t+1|t)$. The target in the training set is, the same as in the training of the NN state predictor, namely, the state $R_{ma}(t+1)$.

The initial number of nodes in the hidden layer is set to 4 and training is done as described in the previous section. Different NN models are developed (with an increasing number of nodes in the hidden layer). The best FMLP NN architecture was found to be a 7-4-1 network, with the average NMSE in the estimation data set (no weight update) at 1.722%. The block diagram for the adaptive NN filter is shown in Figure 55. The armature resistance values of the Motor-Pump process model and NN state filter with the estimation data set as input is shown in Figure 56.

VI.4.3 Hybrid Adaptive Filter for the Armature Resistance and Flux Linkage

Previously, a hybrid adaptive NN filter for the armature resistance was developed. Now, a new filter is designed for the magnetic flux linkage, Ψ , and the armature resistance, $R_a(t)$, simultaneously. The magnetic flux linkage, Ψ , is assumed to be a constant parameter in the process model. In this study, Ψ is modeled as a slowly varying parameter that is, for all practical purposes, it is "constant" for a number of transients, and yet it acquires a new value for another set of transients. Therefore, although Ψ is "constant" for a range of transients, this "constant" value is different.

The hybrid adaptive filter, as discussed earlier, requires two NNs: the NN error model and the NN state filter. The design of these NNs is discussed in the following subsections.

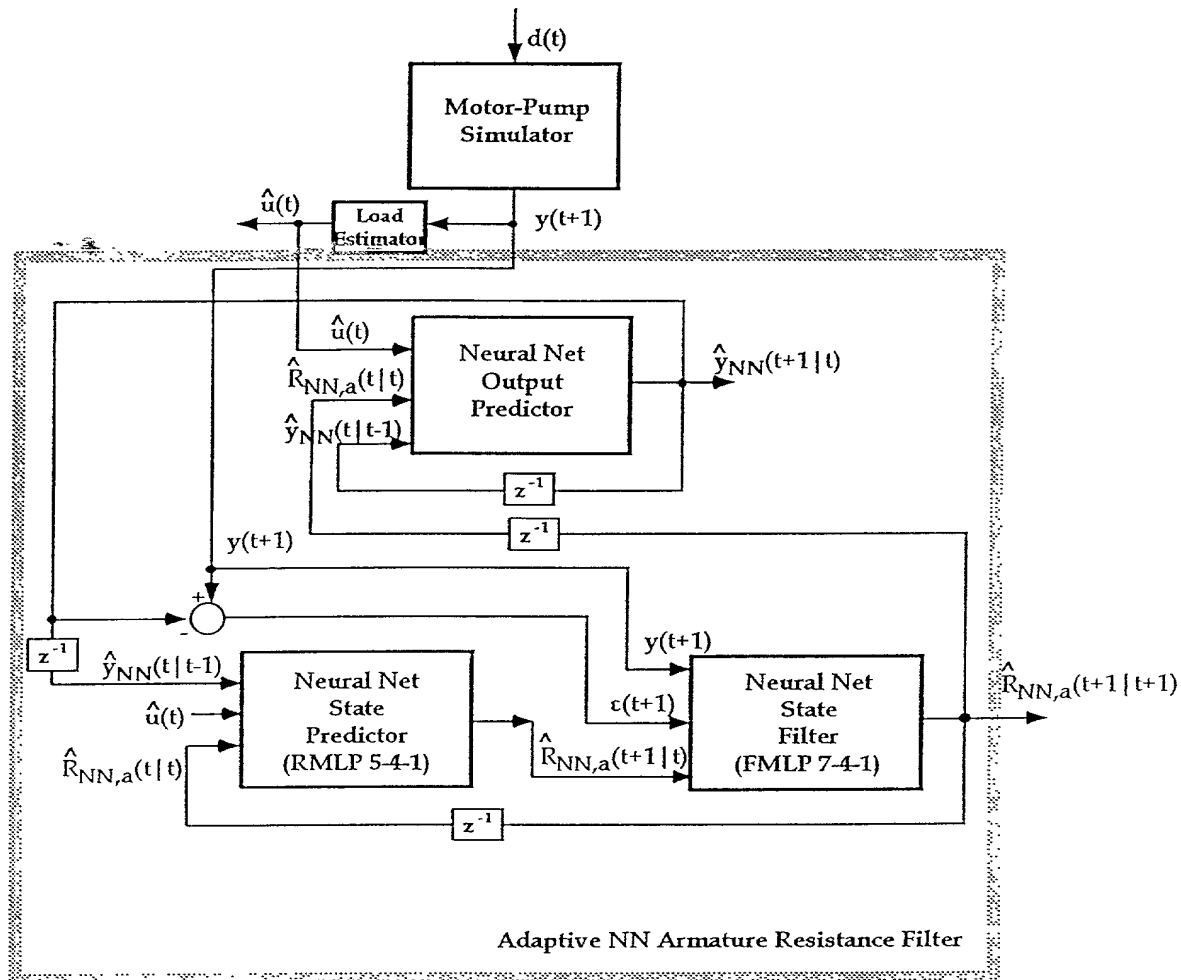


Figure 55. Block Diagram of the Motor-Pump System Adaptive NN Armature Resistance Filter.

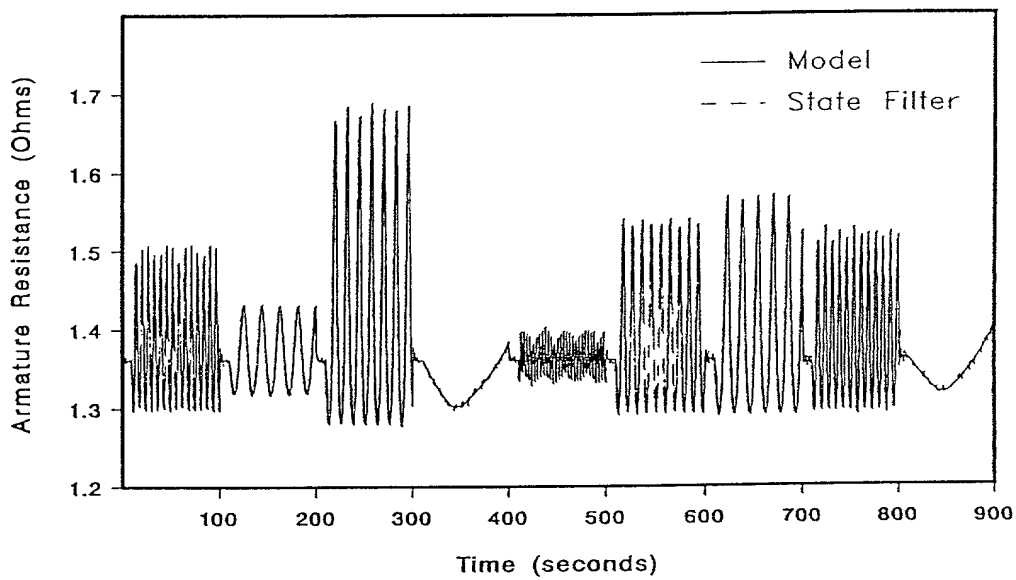
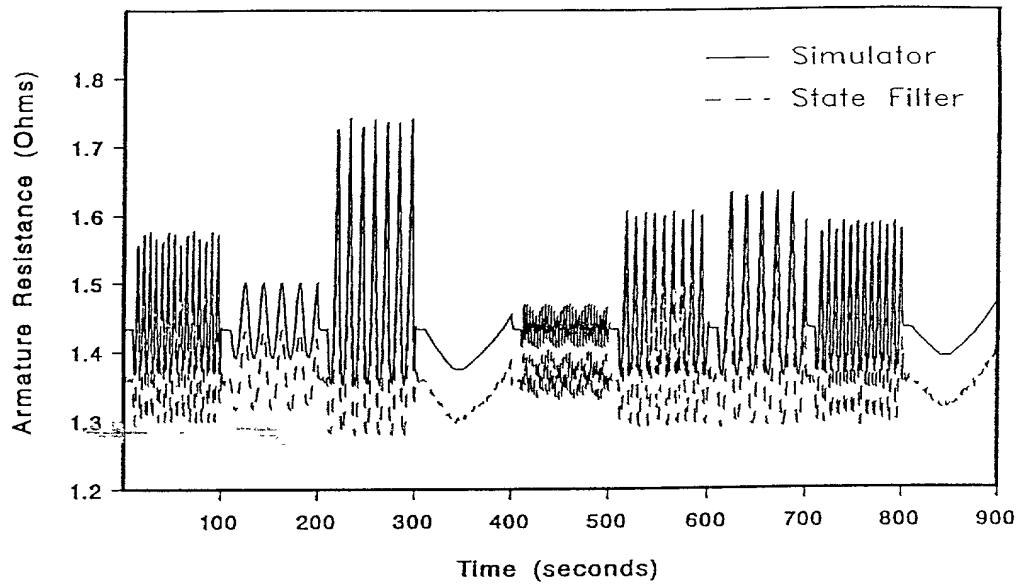


Figure 56. Motor-Pump System Armature Resistance Response, from the Simulator, Process Model, and NN State Filter, for the Adaptive NN Armature Resistance Filter Using the Estimation Data Set as Inputs.

VI.4.3.1 NN Error Model Development

The NN error model is first chosen to be a three-layer FMLP network with 7 nodes in the first layer, corresponding to the 7 inputs: $\hat{u}(t)$, $y(t+1)$ and $\hat{y}_e(t|t)$. The vector $\hat{y}_e(t|t)$ is the difference between the actual process measurements, $y(t)$, and the model predictor output, $\hat{y}(t|t-1)$. The number of nodes in the third (output) layer is set to three (corresponding to the three outputs of $\hat{y}_e(t+1|t+1)$). The number of nodes in the second (hidden) layer is set to 4 initially. The NN Error Model is then trained with the estimation data set. The target is $y_e(t+1)$ which is given by $y(t+1) - \hat{y}(t+1|t)$.

Training is stopped when the NMSE of the NN error model on the estimation data set (no weight update part) just begins to increase (indicating the start of over-training). The NMSE is noted, the number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. The NN architecture with the lowest NMSE in the estimation data set (no weight update) is chosen. In this case, this was a 7-8-3 FMLP NN with a NMSE of 3.9%. The RMLP network was not used to develop the NN error model since the low NMSE obtained by the FMLP network did not warrant the use of the more complex RMLP networks.

VI.4.3.2 NN State Filter Development

The NN state filter architecture is first chosen to be a three-layer FMLP network with 8 nodes in the first layer, corresponding to 8 inputs, and 2 nodes in the third layer, corresponding to 2 outputs. The 8 inputs to the first layer are the three process outputs, $y(t+1)$, the three residuals, $\epsilon(t+1)$ where $\epsilon(t+1) = y(t+1) - \hat{y}_c(t+1|t+1)$, and the states, $R_{ma}(t)$ and Ψ . The vector $\hat{y}_c(t+1|t+1)$ is the *corrected* process model predictor output, and it is the sum of the model predictor output, $\hat{y}(t+1|t)$, and the output of the NN error model, $\hat{y}_e(t+1|t+1)$. The two outputs are $\hat{R}_{NN,a}(t+1|t+1)$ and $\hat{\Psi}_{NN}(t+1|t+1)$. Initially, the number of nodes in the hidden layer is chosen to be 3. The NN state filter is then trained with the estimation data set. The targets

are the model resistance, $R_{ma}(t+1)$, and magnetic flux linkage, Ψ , as determined by the process model.

Training is stopped when the NMSE of the estimation data set (no weight update) just begins to increase (indicating the start of over-training). The NMSE is noted, the number of nodes in the hidden layer is increased, and the training cycle is repeated. In this case, it was determined that the FMLP NN with architecture 8-9-2 was the best network. The NMSE of this network outputs on the estimation data set (no weight update) is 0.5%. A block diagram of the hybrid adaptive filter is shown Figure 57. A comparison between the Motor-Pump simulator outputs, $y(t)$, and the corrected model predictor outputs, $\hat{y}_c(t|t)$, is shown in Figure 58. The comparison between the simulator state values, $R_a(t)$ and $\Psi(t)$ and the NN state filter outputs, $\hat{R}_{NN,a}(t|t)$ and $\Psi_{NN}(t|t)$, is shown in Figure 59.

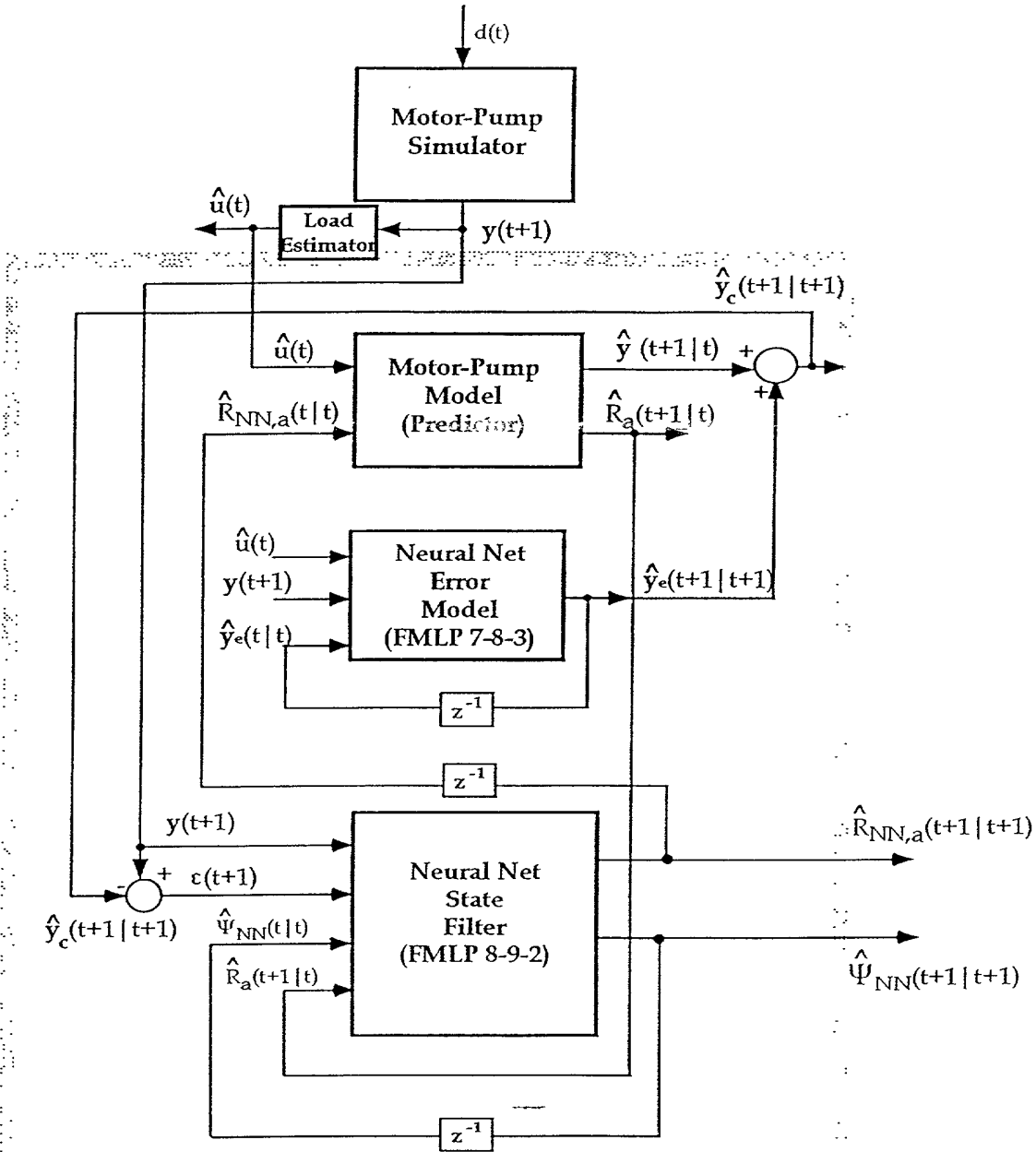
The RMLP network was not used to develop the NN state filter since the FMLP network performed adequately, and the additional complexity of the RMLP network did not warrant its use.

VI.5 Validation Results

The hybrid adaptive and the adaptive state filters developed in the previous sections are now evaluated using the validation data sets. The validation data set used in evaluating the armature resistance filters, hybrid adaptive and adaptive, is listed in Table 4. The validation data set used in evaluating the hybrid adaptive armature resistance and flux linkage filter is listed in Table 5.

Each test signal in the validation data set is augmented with zero-mean, white, Gaussian noise with 0.02 sd (low noise), and the performance of the state filters is evaluated. The test signals are then augmented with zero-mean, white, Gaussian noise of 0.08 sd and the state filters are again evaluated under the higher noise conditions.

The performance of the state filters developed in the previous sections are described in the following subsections.



Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter

Figure 57. Block Diagram of the Motor-Pump System Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter.

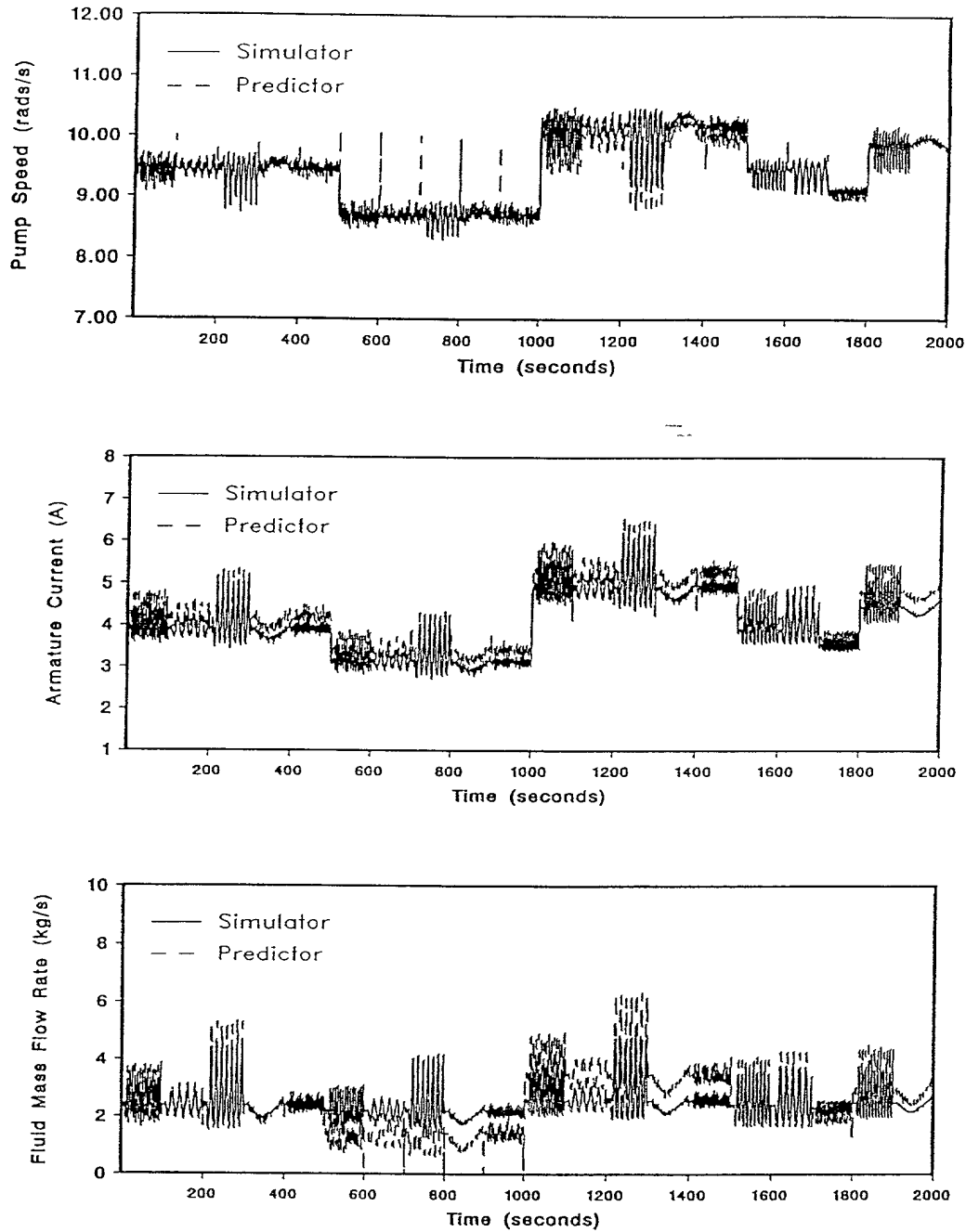


Figure 58. Motor-Pump System Output Response, from the Simulator and Model Predictor, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Estimation Data Set as Inputs.

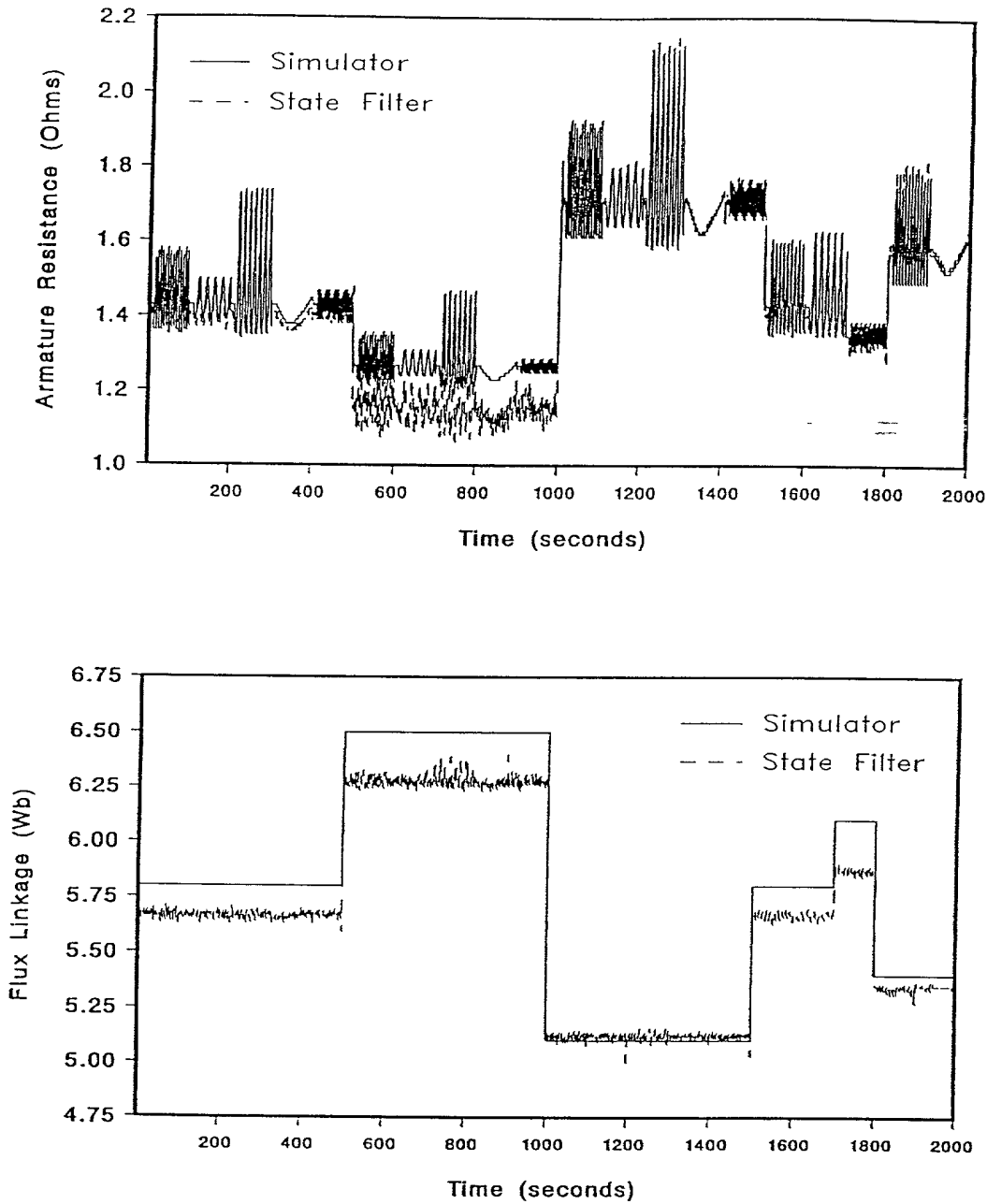


Figure 59. Motor-Pump System Armature Resistance and Flux Linkage Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Estimation Data Set as Inputs.

VI.5.1 Hybrid Adaptive Filter for the Armature Resistance

The hybrid adaptive armature resistance filter is evaluated by subjecting the system to the validation data set. The Motor-Pump model predictor (corrected) transient response is shown in Figure 60. The average NMSE for the state filtered value, $\hat{R}_{NN,a}(t|t)$, is 0.34%. The state estimate values for the validation data set is shown in Figure 61. From Figure 61, it is seen that the filtered state value, $\hat{R}_{NN,a}(t|t)$, will always show a discrepancy when compared with the process simulator state value, $R_a(t)$. This is because the NN state filter was trained with the Motor-Pump process model state values, $R_{ma}(t)$, and *not* with the process simulator state values, $R_a(t)$. Therefore, the NN state filter values, $\hat{R}_{NN,a}(t|t)$, will only be as accurate as the Motor-Pump Model state values, $R_{ma}(t)$. This is further shown in Figure 62 where the normalized residuals, $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$, are plotted. The average $\epsilon_{Sim}(t)$ value is about 4.8% while the average $\epsilon_{Mod}(t)$ value is about 0.3%.

The performance of the hybrid adaptive state filter is then evaluated with the same validation data set but with higher process and measurement noise (zero-mean, white, Gaussian with 0.08 sd). The Motor-Pump model predictor (corrected) transient response is shown in Figure 63. The average NMSE for the state filter value, $\hat{R}_a(t|t)$, is 0.38%. The filtered state values for the validation data set is shown in Figure 64. The residuals $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$ are plotted in Figure 65. The average $\epsilon_{Sim}(t)$ value is about 5.4% while the average $\epsilon_{Mod}(t)$ value is about 0.2%.

VI.5.2 Adaptive Filter for the Armature Resistance

The adaptive state filtering method, developed earlier, is evaluated using the validation data set. The NN output predictor, including three RMLP 5-6-1 networks, transient response is shown in Figure 66. The average NMSE for the filtered state value, $\hat{R}_{NN,a}(t|t)$, is 0.33%. The filtered state values for the validation data set is shown in Figure 67. From Figure 67, it is seen that the filtered state value, $\hat{R}_{NN,a}(t|t)$, will always show a discrepancy when compared to the actual process state value, $R_a(t)$. This is because the NN state filter was trained with the Motor-Pump process

509

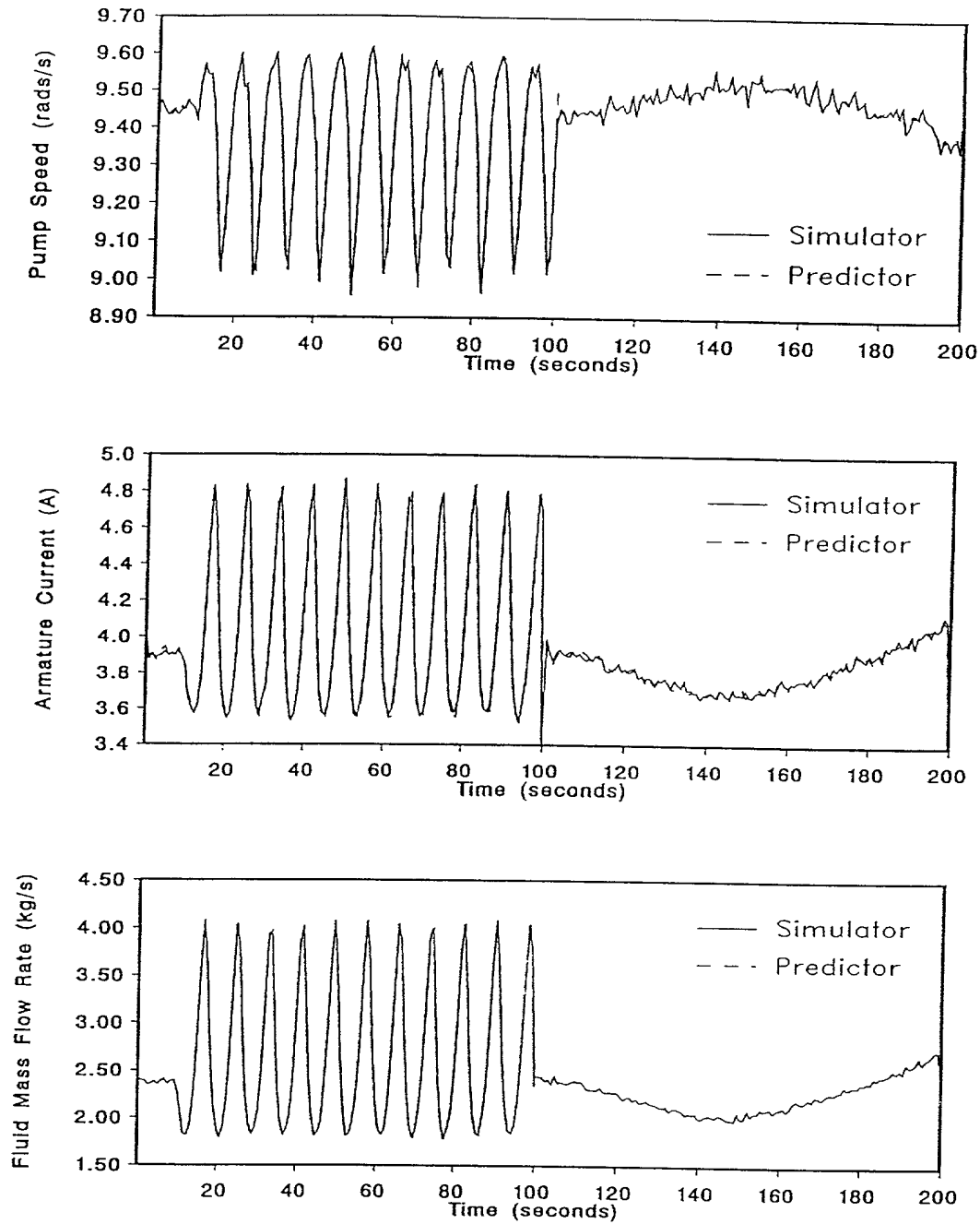


Figure 60. Motor-Pump System Output Response, from the Simulator and the Model Predictor, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).

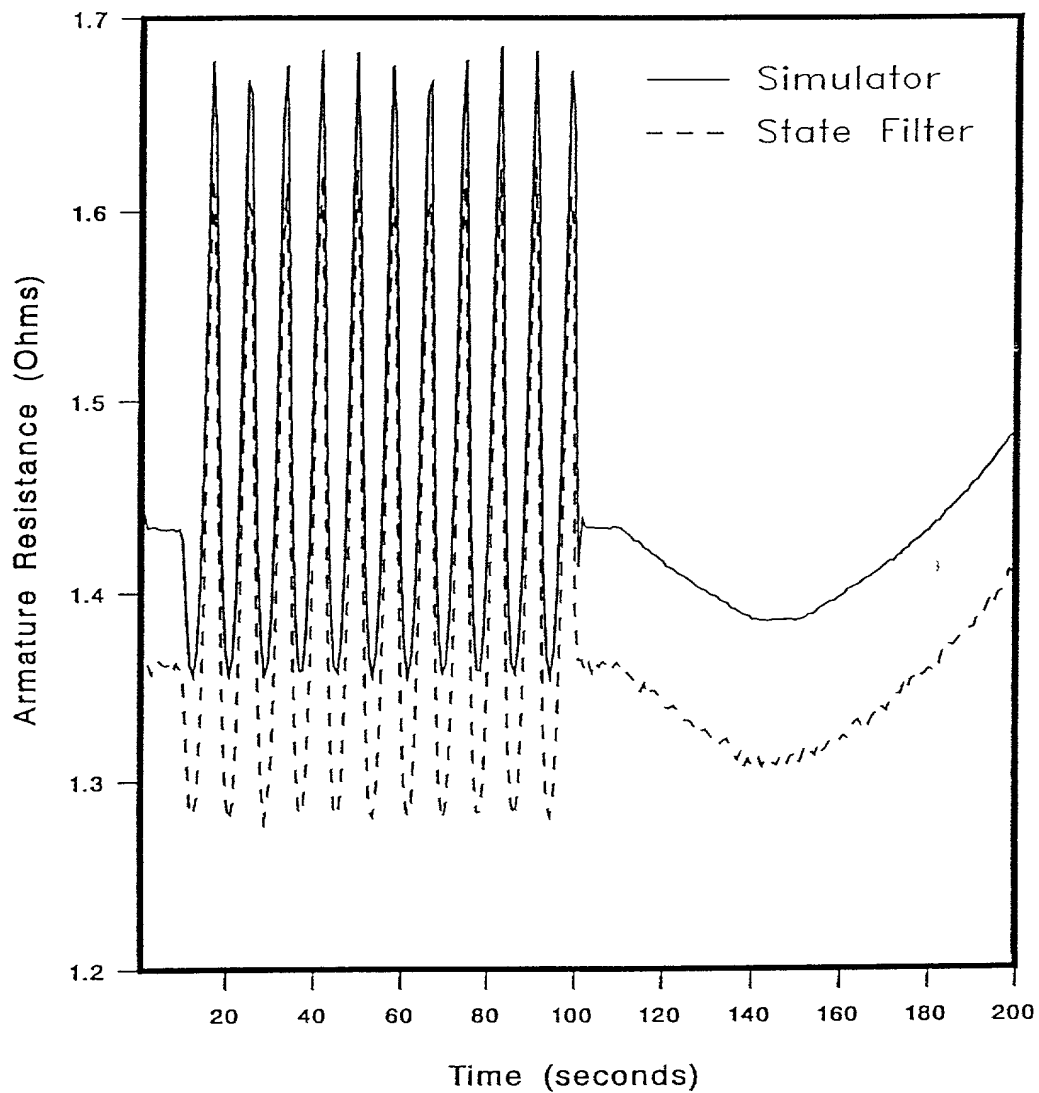


Figure 61. Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).

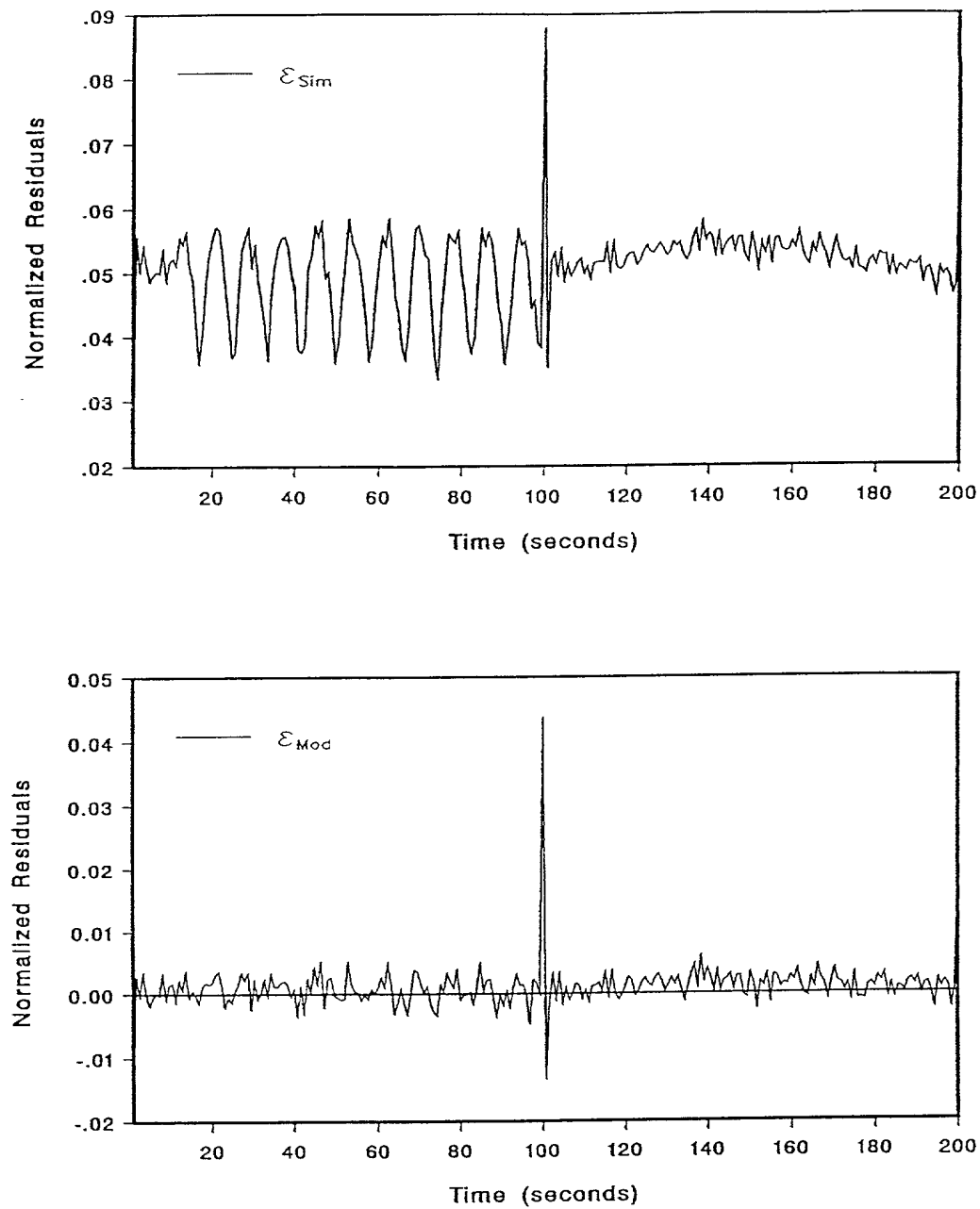


Figure 62. Motor-Pump System Armature Resistance Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).

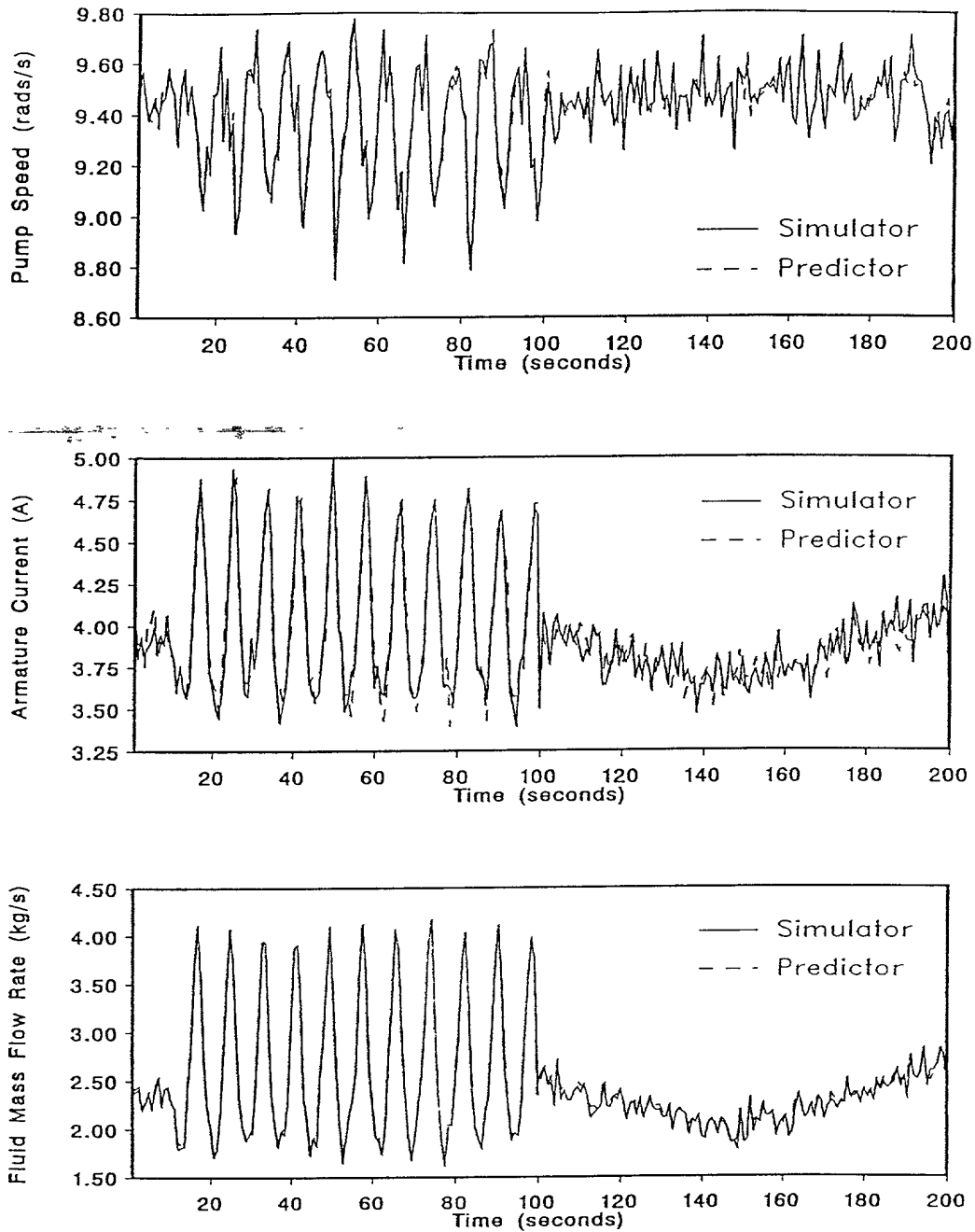


Figure 63. Motor-Pump System Output Response, from the Simulator and the Model Predictor, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).

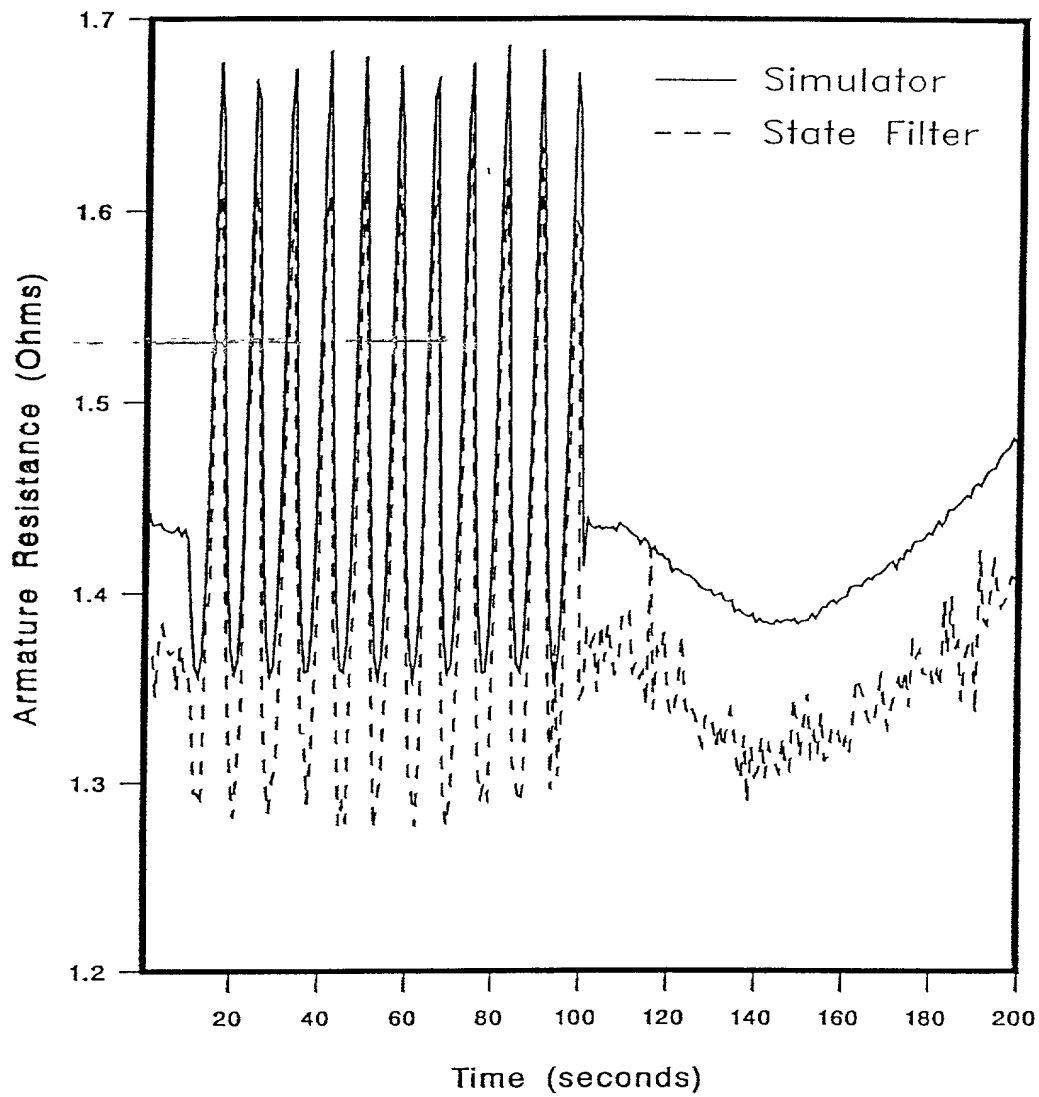


Figure 64. Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).

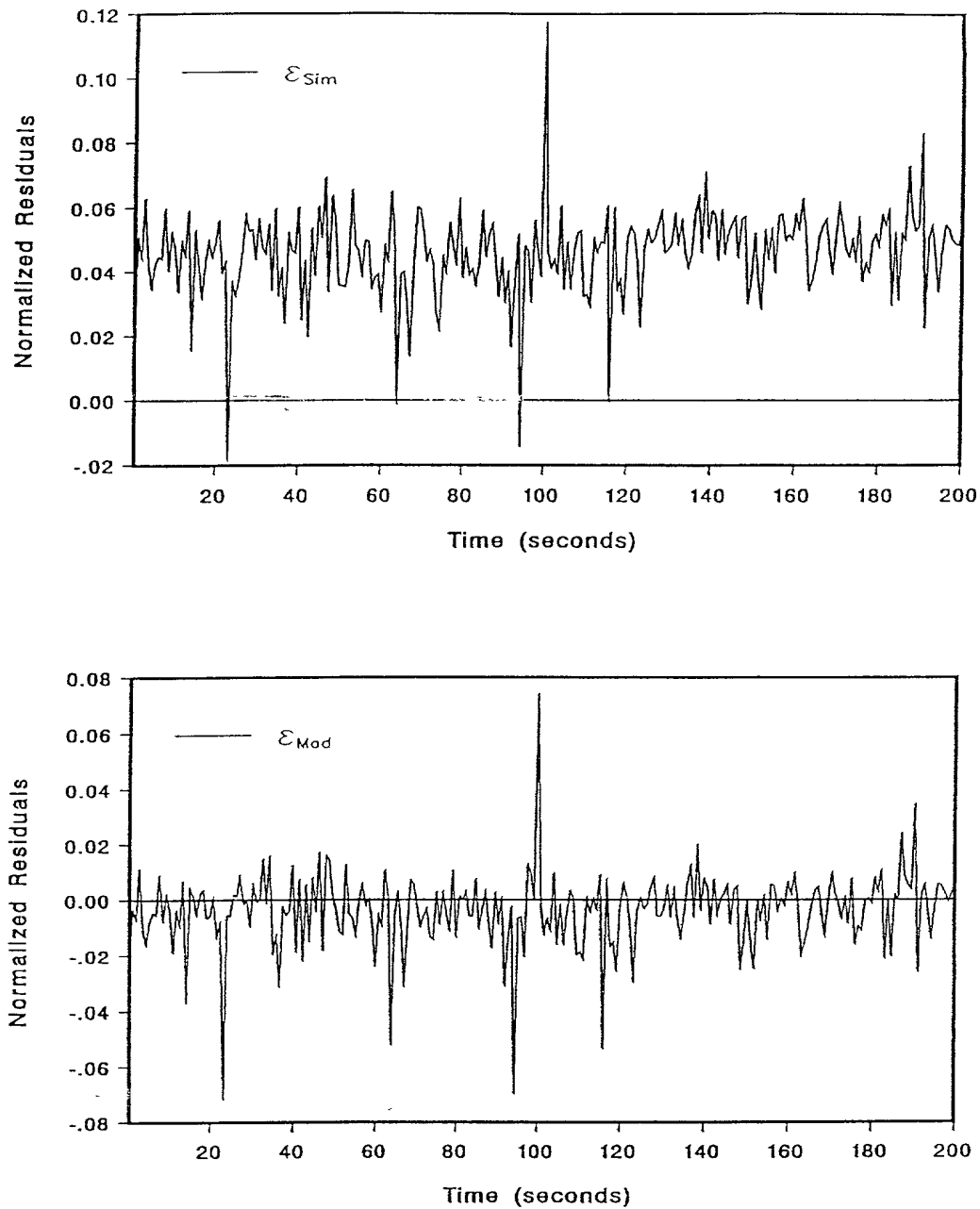


Figure 65. Motor-Pump System Armature Resistance Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).

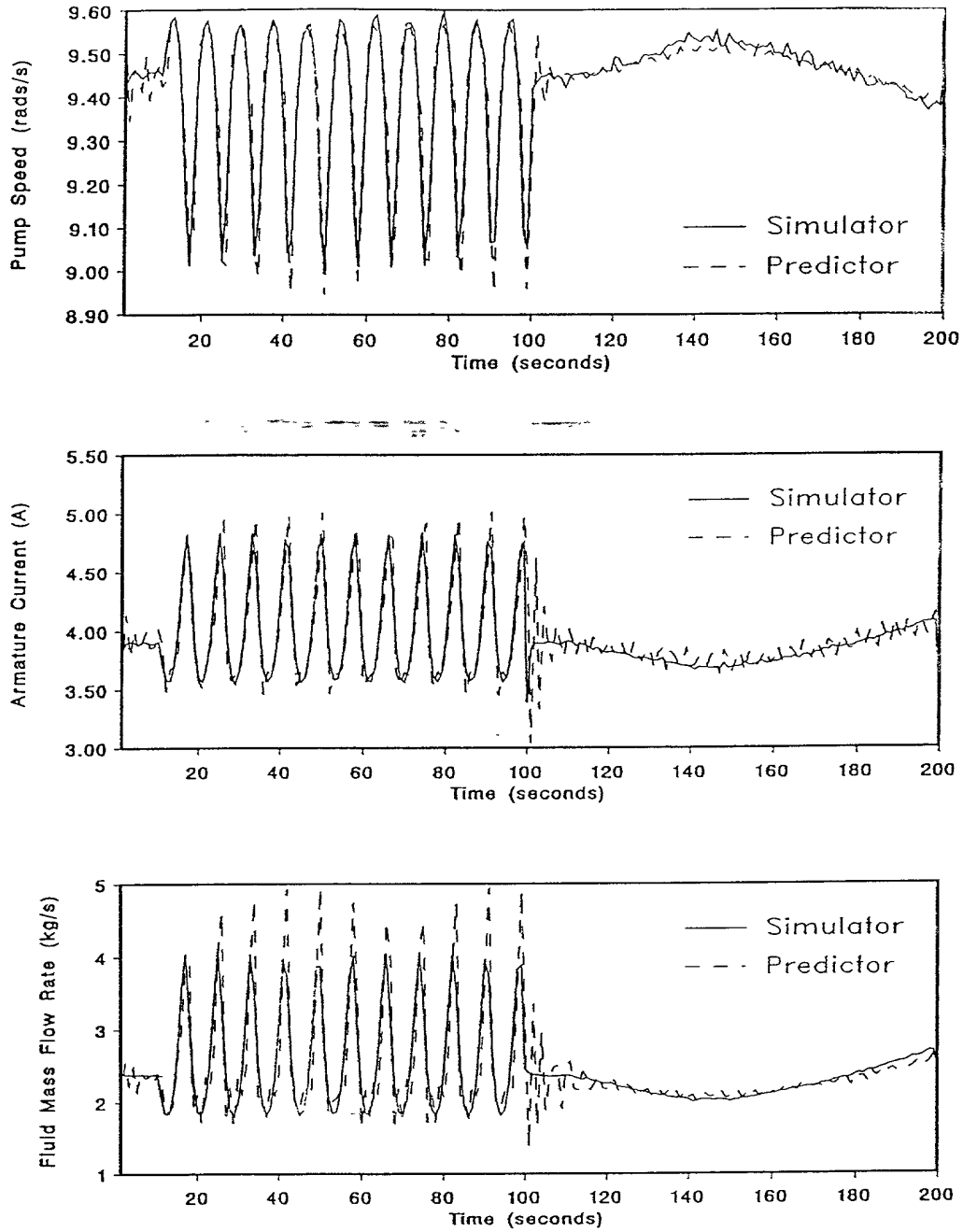


Figure 66. Motor-Pump System Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).

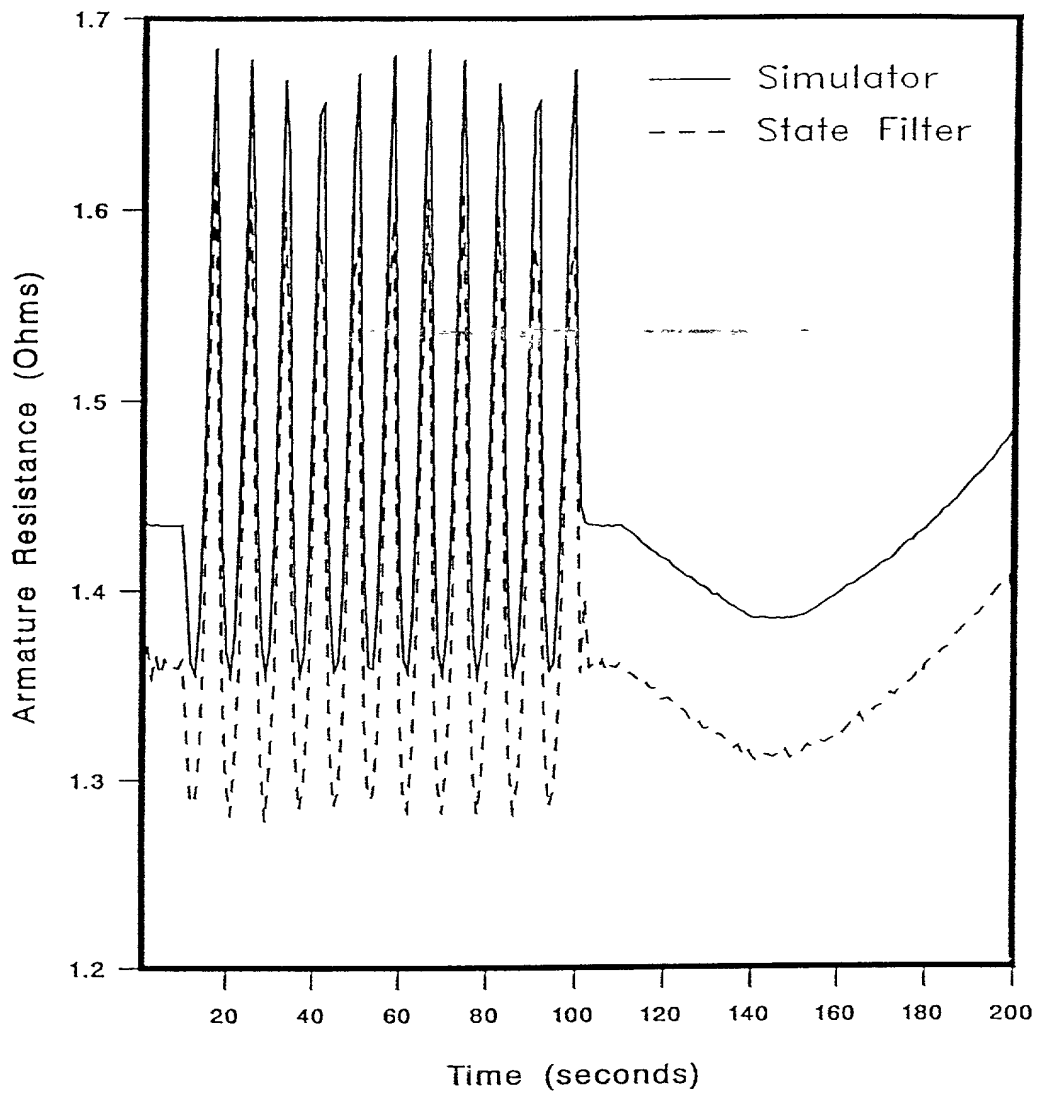


Figure 67. Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).

model state values, $R_{ma}(t)$, and *not* with the process simulator state values, $R_a(t)$. Therefore, the NN state filter values, $\hat{R}_{NN,a}(t|t)$, will only be as accurate as the Motor-Pump model state values, $R_{ma}(t)$. This is further seen in Figure 68 where the normalized residuals $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$ are plotted. The average $\epsilon_{Sim}(t)$ value is about 4.9% while the average $\epsilon_{Mod}(t)$ value is about 0.45%.

The performance of the adaptive filter is then evaluated with the same validation data set but with higher process and measurement noise (zero-mean, white, Gaussian with 0.08 sd). The NN output predictor transient response is shown in Figure 69. The average NMSE for the filtered state value, $\hat{R}_{NN,a}(t|t)$, is 0.42%. The filtered state values for the validation data set is shown in Figure 70. The residuals $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$ are plotted in Figure 71. The average $\epsilon_{Sim}(t)$ value is about 5.3% while the average $\epsilon_{Mod}(t)$ value is about 0.5%.

VI.5.3 Hybrid Adaptive Filter for the Armature Resistance and Flux Linkage

The hybrid adaptive armature resistance and flux linkage NN filter is evaluated using the validation data set represented in Table 5. The average NMSE for the filtered state values, $\hat{R}_{NN,a}(t|t)$ and $\hat{\Psi}_{NN}(t|t)$, is 0.12% and 0.1%, respectively. The filtered state values for the validation data set are shown in Figure 72. In this case, since the hybrid adaptive NN filter was developed with data from a process model that is very close to the process simulator, as discussed earlier. Therefore, the filter performance is evaluated using only the $\epsilon_{Sim}(t)$ values, and not both the $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$ values as was done in the NN filters for the armature resistance discussed earlier. The residuals $\epsilon_{Sim}(t)$ for both the armature resistance and flux linkage filtered values are shown in Figure 73. The average $\epsilon_{Sim}(t)$ value for the filtered value $\hat{R}_{NN,a}(t|t)$ is 1.5% while the average $\epsilon_{Sim}(t)$ value for the filtered value $\hat{\Psi}_{NN}(t|t)$ is 2.3%.

The performance of the hybrid adaptive armature resistance and flux linkage filtering method is then evaluated with the same validation data set but with higher

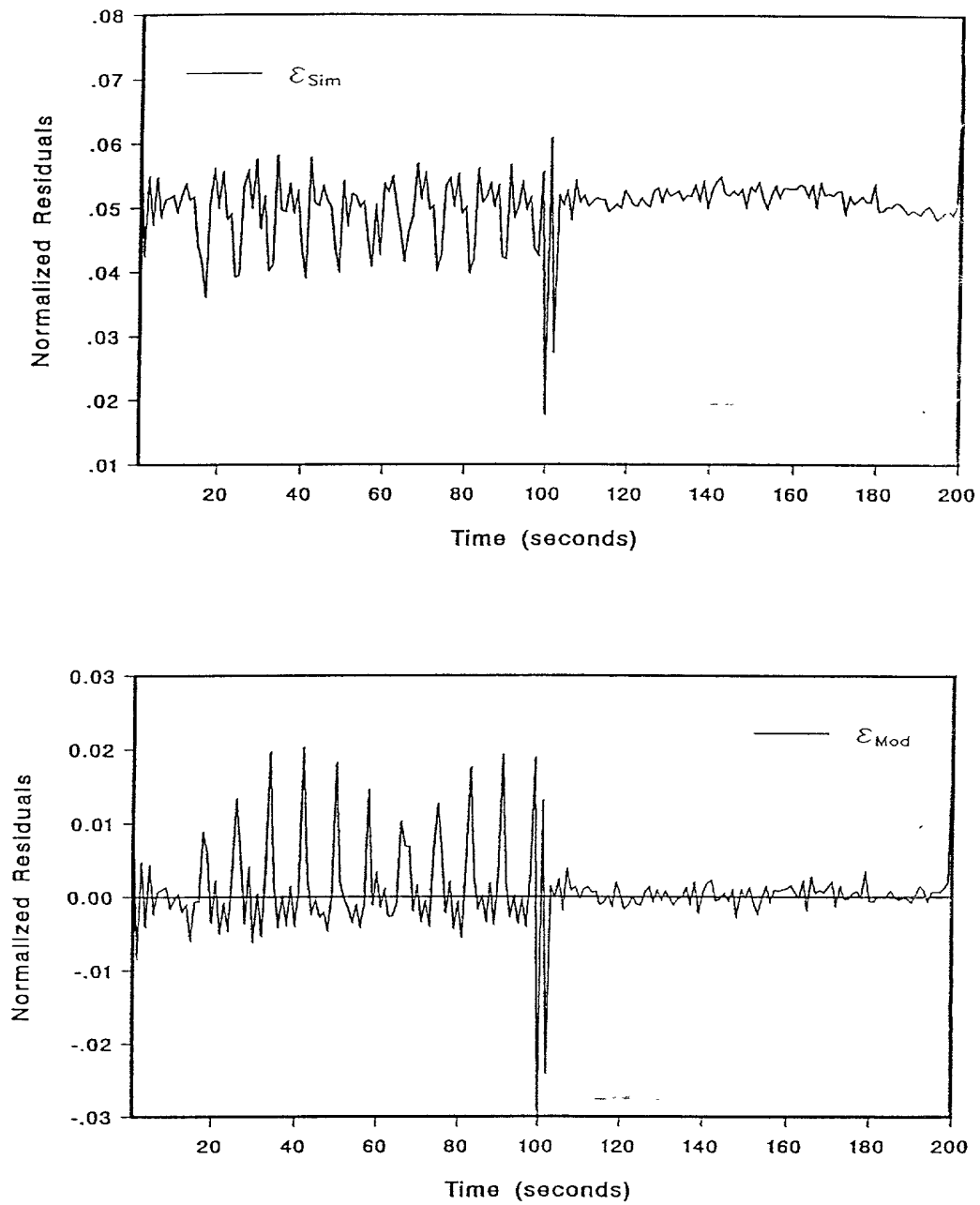


Figure 68. Motor-Pump System Armature Resistance Filter Normalized Residuals for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (Low Noise Environment).

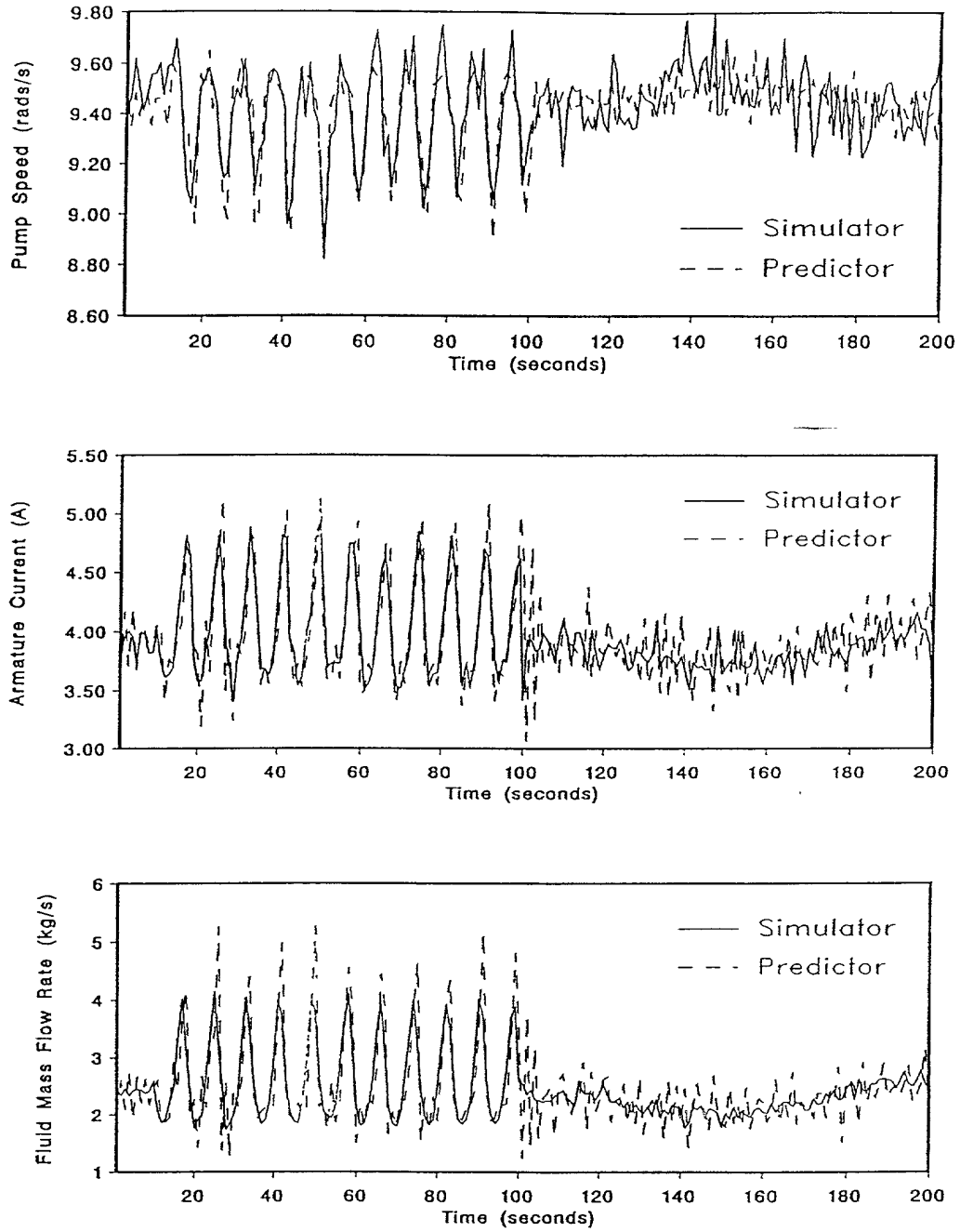


Figure 69. Motor-Pump System Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).

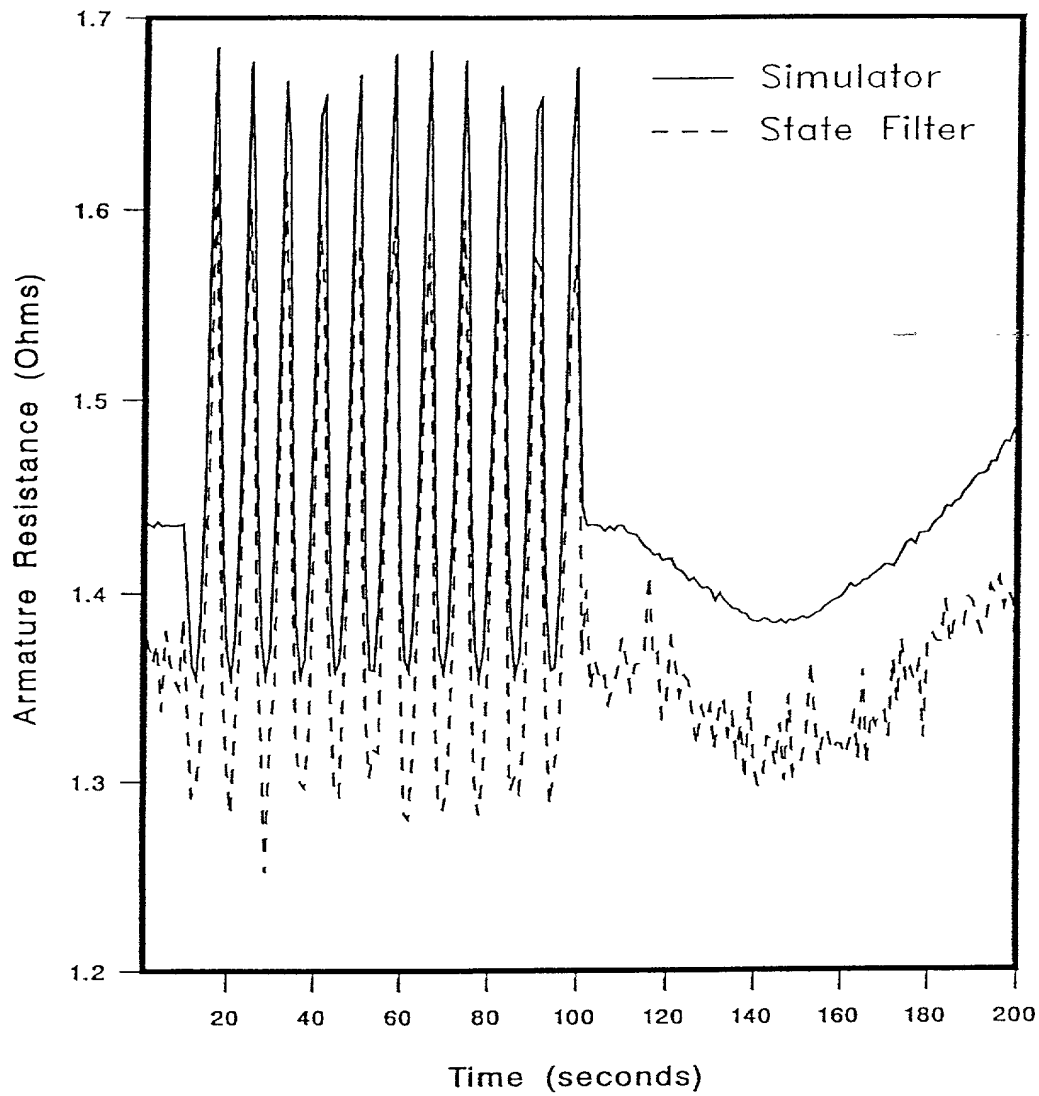


Figure 70. Motor-Pump System Armature Resistance Response, from the Simulator and NN State Filter, for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).

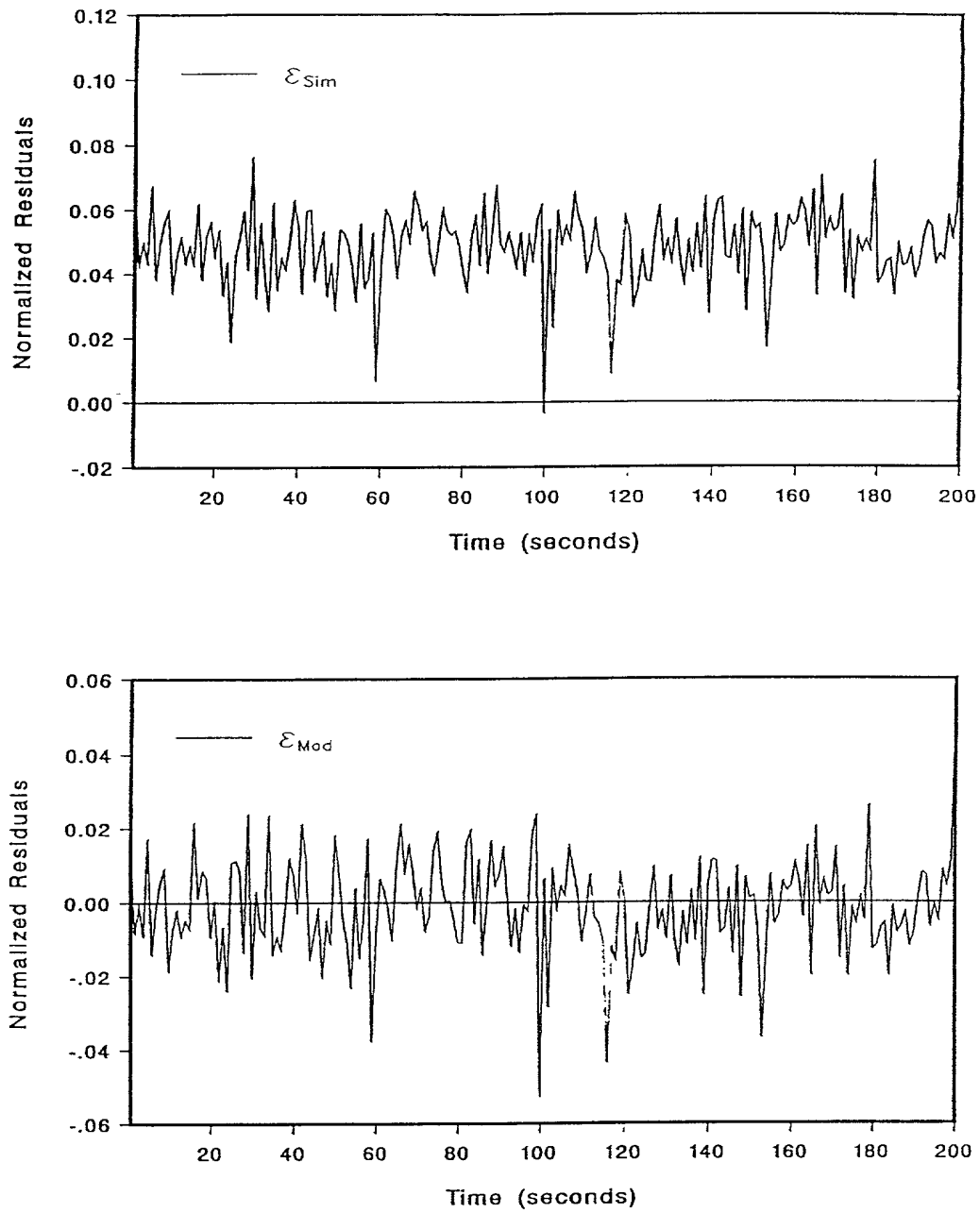


Figure 71. Motor-Pump System Armature Resistance Filter Normalized Residuals for the Adaptive NN Armature Resistance Filter Using the Validation Data Set as Inputs (High Noise Environment).

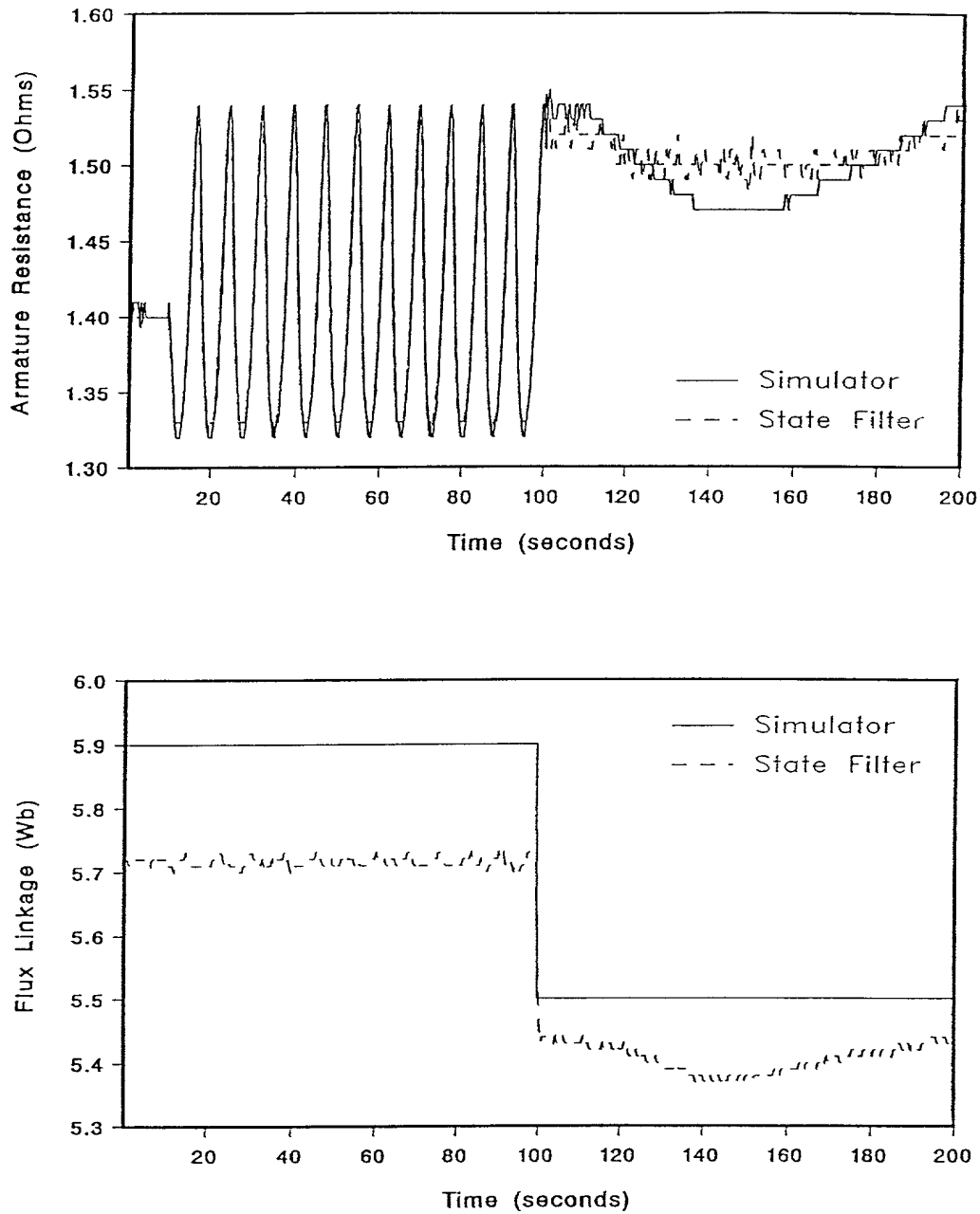


Figure 72. Motor-Pump System Armature Resistance and Flux Linkage Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (Low Noise Environment).

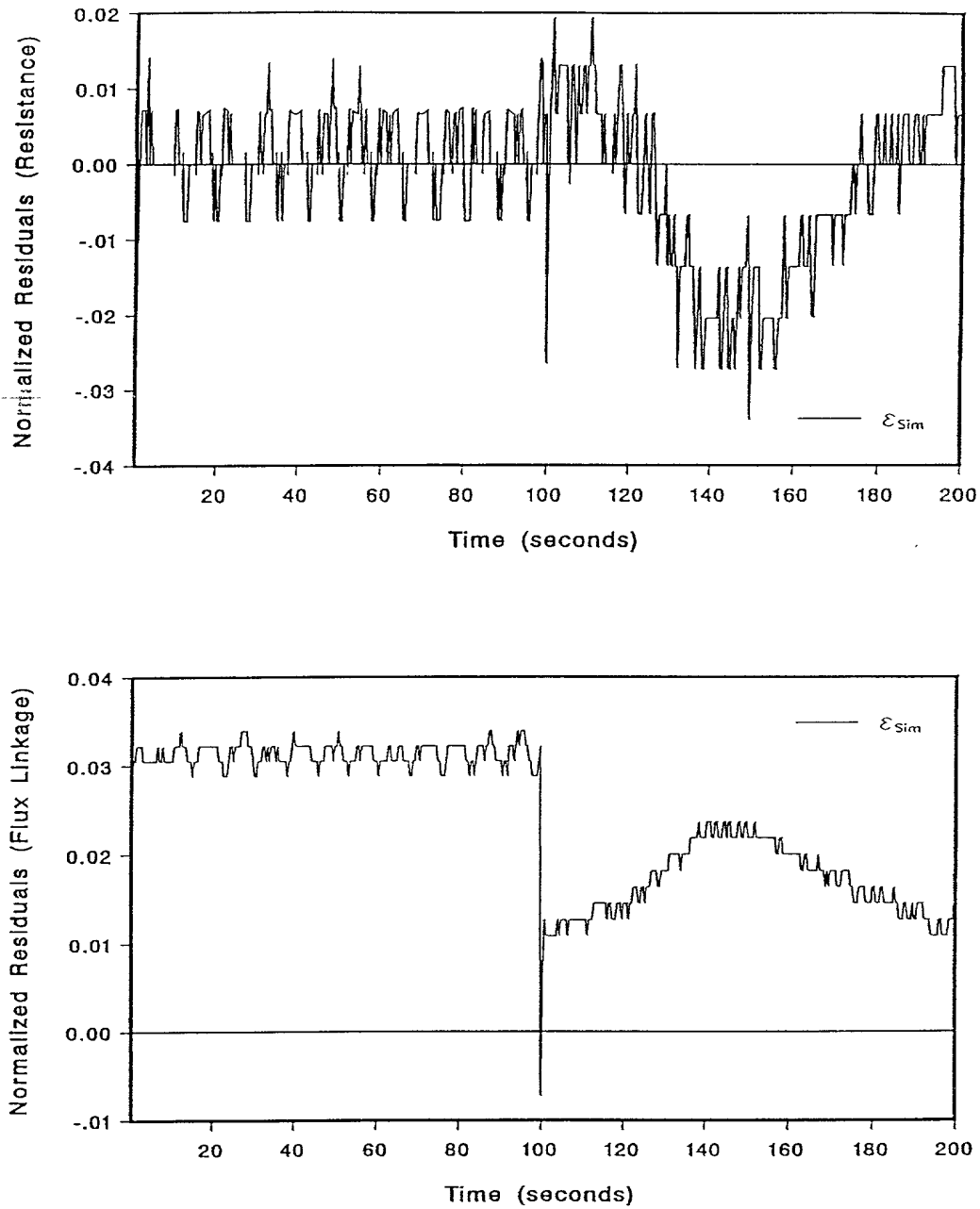


Figure 73. Motor-Pump System Armature Resistance and Flux Linkage Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (Low Noise Environment).

process and measurement noise (zero-mean, white, Gaussian with 0.08 sd) added to it. The average NMSE for the state filtered values, $\hat{R}_{NN,a}(t|t)$ and $\hat{\Psi}_{NN}(t|t)$, is 0.22% and 0.18%, respectively. The state estimate values for the validation data set are shown in Figure 74. The residuals $\epsilon_{\text{Sim}}(t)$ for both $\hat{R}_{NN,a}(t|t)$ and $\hat{\Psi}_{NN}(t|t)$ are plotted in Figure 75. The average $\epsilon_{\text{Sim}}(t)$ value for $\hat{R}_{NN,a}(t|t)$ is 4.3% while the average $\epsilon_{\text{Sim}}(t)$ value for $\hat{\Psi}_{NN}(t|t)$ is 2.6%.

VI.6 Chapter Summary

A Motor-Pump process system was used to develop hybrid adaptive and adaptive NN armature resistance filters. Additionally, the flux linkage, a constant parameter in the Motor-Pump process, is also filtered using an hybrid adaptive filter. The Motor-Pump system is represented by three states: the angular velocity of the motor/pump, $\omega(t)$; the armature current, $I_a(t)$; and the fluid mass flow rate, $\dot{M}(t)$. These states are also the measured outputs of the process. The single input disturbance is the variation in valve resistance, caused by the valve opening and closing, which is not measurable. Therefore, the effect of the process input disturbance is modeled as a load term, which is estimated by a semi-empirical relationship between two of the measurements, $\omega(t)$ and $\dot{M}(t)$. The estimation and validation data set is a collection of data corresponding to sinusoids of different amplitudes and frequencies, and ramps with different slopes, as the input disturbance.

A hybrid adaptive and an adaptive NN filter is developed to estimate the unmeasurable process state, the armature resistance $R_a(t)$. In addition, a hybrid adaptive filter is designed to estimate the armature resistance and magnetic flux linkage, Ψ , simultaneously. The developed filters are then evaluated with an independent data set, the validation data set. The results of these tests are summarized in Tables 6 and 7. Note that in Table 7 only the average $\epsilon_{\text{Sim}}(t)$ value is reported. This is done since the process model, used in the development of the hybrid adaptive NN armature resistance and flux linkage filter, is very close to the simulator. Therefore, $\epsilon_{\text{Mod}}(t)$ is very close to $\epsilon_{\text{Sim}}(t)$.

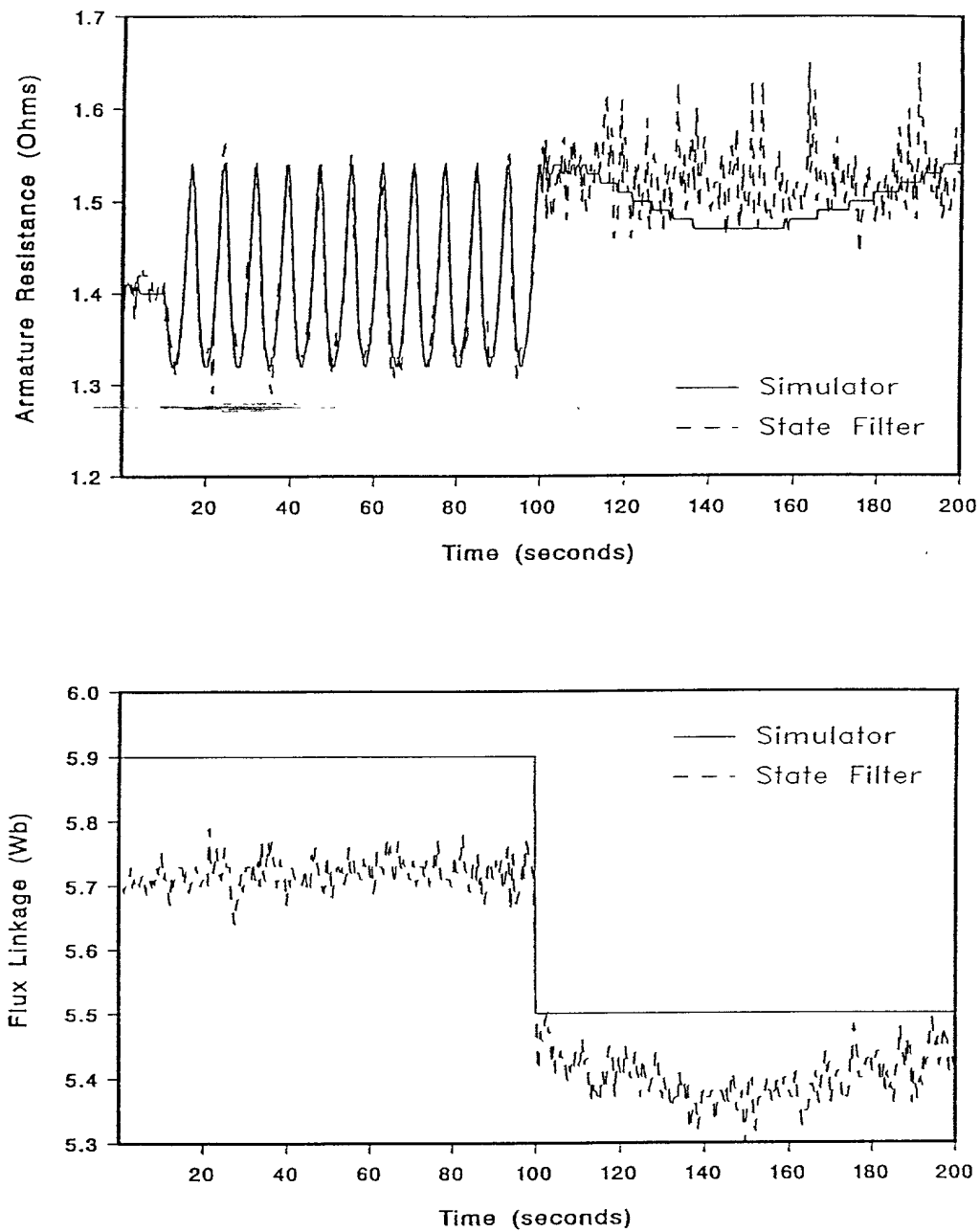


Figure 74. Motor-Pump System Armature Resistance and Flux Linkage Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (High Noise Environment).

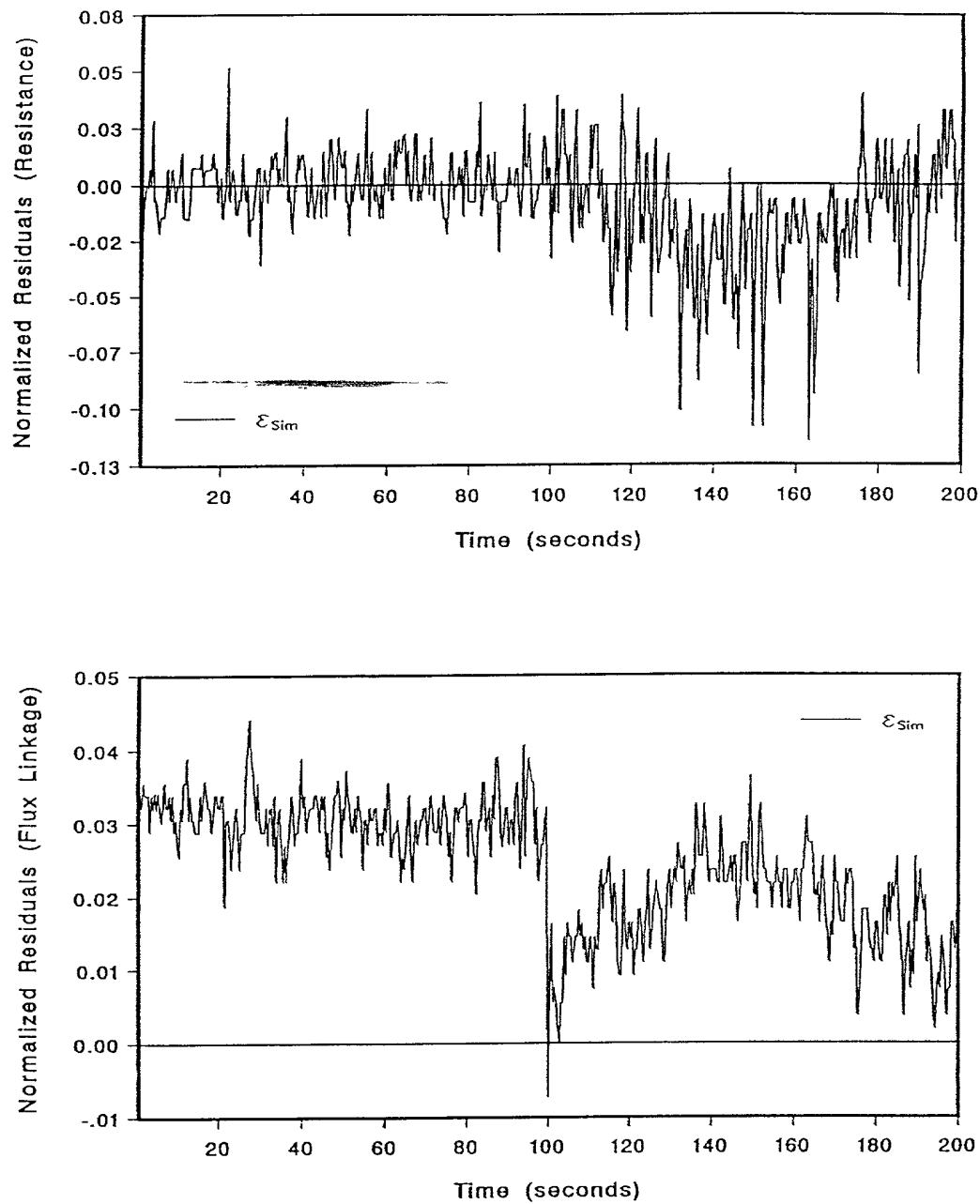


Figure 75. Motor-Pump System Armature Resistance and Flux Linkage Filter Normalized Residuals for the Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Using the Validation Data Set as Inputs (High Noise Environment).

Table 6. Motor-Pump System Hybrid Adaptive and Adaptive NN Armature Resistance Filter Validation Test Results.

NN State Filter		$\overline{\text{NMSE}}$	$\overline{\epsilon_{\text{Sim}}(t)}$	$\overline{\epsilon_{\text{Mod}}(t)}$
Hybrid Adaptive	Low Noise ($\sigma = 0.02$)	0.34%	4.8%	0.3%
	High Noise ($\sigma = 0.08$)	0.38%	5.4%	0.2%
Adaptive	Low Noise ($\sigma = 0.02$)	0.33%	4.9%	0.45%
	High Noise ($\sigma = 0.08$)	0.42%	5.3%	0.5%

Table 7. Motor-Pump System Hybrid Adaptive NN Armature Resistance and Flux Linkage Filter Validation Test Results.

NN State Filter		$\overline{\text{NMSE}}$		$\overline{\epsilon_{\text{Sim}}(t)}$	
		$\hat{R}_a(t t)$	$\hat{\Psi}(t t)$	$\hat{R}_a(t t)$	$\hat{\Psi}(t t)$
Hybrid Adaptive	Low Noise ($\sigma = 0.02$)	0.12%	0.1%	1.5%	2.3%
	High Noise ($\sigma = 0.08$)	0.22%	0.18%	4.3%	2.6%

It is observed, from the validation test results, that the overall performance of both the hybrid adaptive and adaptive armature resistance NN filters is good. Furthermore, the accuracy of the armature resistance filter is not affected much by high noise environment. This can be attributed to the noise rejection properties of the NNs. The results of the tests on the armature resistance and flux linkage NN filter indicate that states with a wide range of dynamic time constants can be effectively filtered.

CHAPTER VII

APPLICATION 3 - U-TUBE STEAM GENERATOR SYSTEM

VII.1 Introduction

In this chapter, a U-Tube Steam Generator (UTSG) process system is used to develop the NN state filters based on the methods discussed in Chapter IV. The UTSG is a highly nonlinear, unstable, multivariable process system used in nuclear power plants using pressurized water reactors (PWRs). The UTSG is used to convert water in liquid phase to steam. The steam from the UTSG is used to drive turbines which in turn drive electrical generators. The UTSG is, therefore, an important power plant component whose operation is critical for the overall plant operation. The UTSG process system is sufficiently complex that it is considered a “real-world” for testing the NN state filters previously developed.

A plant validated UTSG simulator is used to develop and test hybrid adaptive and adaptive NN state filters. Two UTSG process models are used, separately, in the development of the hybrid adaptive state filters: a scheduled model based on several linear UTSG models, and the simulator itself without process noise. The scheduled UTSG process model is fairly reliable, but it is still much less accurate than the simulator process model. The state that is filtered is the riser void fraction, $\alpha_R(t)$, which cannot be measured on-line. Additionally, an adaptive filter is designed for filtering a slowly varying parameter: the overall primary heat transfer coefficient, U_{over} .

This chapter is organized into 5 sections: Section VII.2 gives a detailed description of the UTSG process simulator. Section VII.3 gives a description of the UTSG models used in the development of the NN state filters. Section VII.4 describes the development of the NN state filters for the UTSG. Section VII.5 documents the

validation test results performed on the NN state filters. The final section, VII.6, summarizes the chapter.

VII.2 Process Description

A schematic diagram of a Westinghouse type U-Tube Steam Generator (UTSG) is shown in Figure 76. High pressure liquid from the plant primary loop flows in and out of the steam generator primary side (tube side). This is indicated in the figure as *Hot Leg In* and *Cold Leg Out*. The ~~primary-side liquid flow~~ path is up from the hot leg inlet, to a semicircular turnaround, and then down to the cold-leg exit. Feedwater enters the UTSG from the steam plant, as indicated by the arrow *Feedwater In*, through a normally submerged feed ring. At this junction, the feedwater mixes with the liquid being discharged from the liquid-vapor separation devices. The liquid mixture flows downward through the annular *Downcomer* region. After a turn at the bottom of the Downcomer, the liquid is heated during an upward passage outside the U-Tubes in the *Tube Bundle* region. The heat transferred across the tube bundles causes evaporation of some of the liquid, and a two-phase mixture exits from the tube bundle entering the *Riser*. The liquid and vapor are partially separated by swirl vanes at the top of the riser and more separation occurs near the top of the UTSG. The vapor that results is indicated by *Steam Out*, and is saturated with a quality of about 99.8%. This vapor is fed to the turbine units and other auxiliary equipment [79].

An existing UTSG simulator developed by Strohmayer [74], and modified by Choi [15], was adopted for the purpose of this study. The simulator was developed using one-dimensional mass, momentum, and energy conservation equations. An integrated secondary-recirculation-loop momentum equation is incorporated into the simulator to calculate the water level. Because the open-loop system is unstable, a stabilizing controller is required to allow system operation. The controller structure used with the above UTSG simulator is that proposed by Menon and Parlos [48].

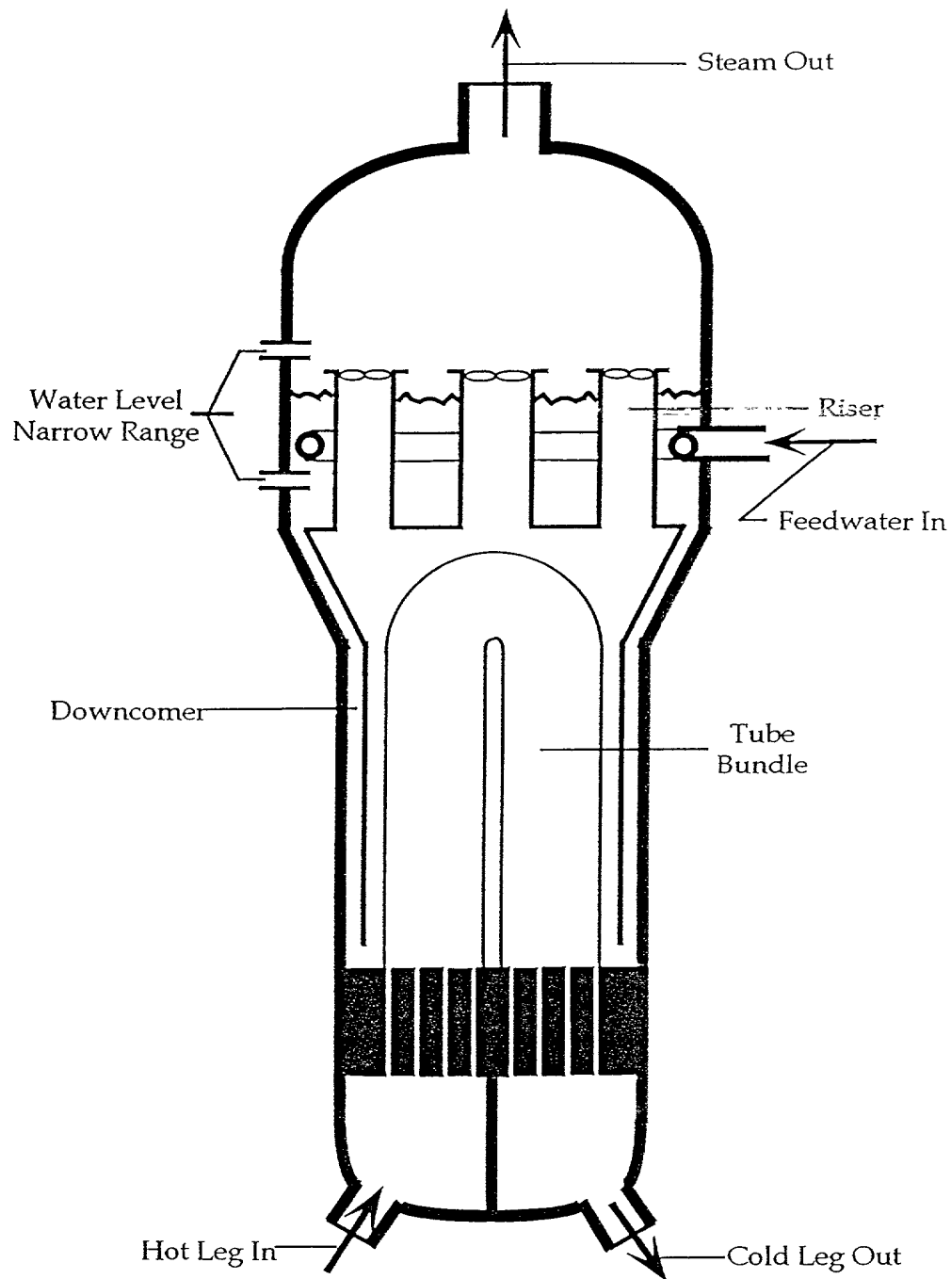


Figure 76. Schematic Diagram of the U-Tube Steam Generator.

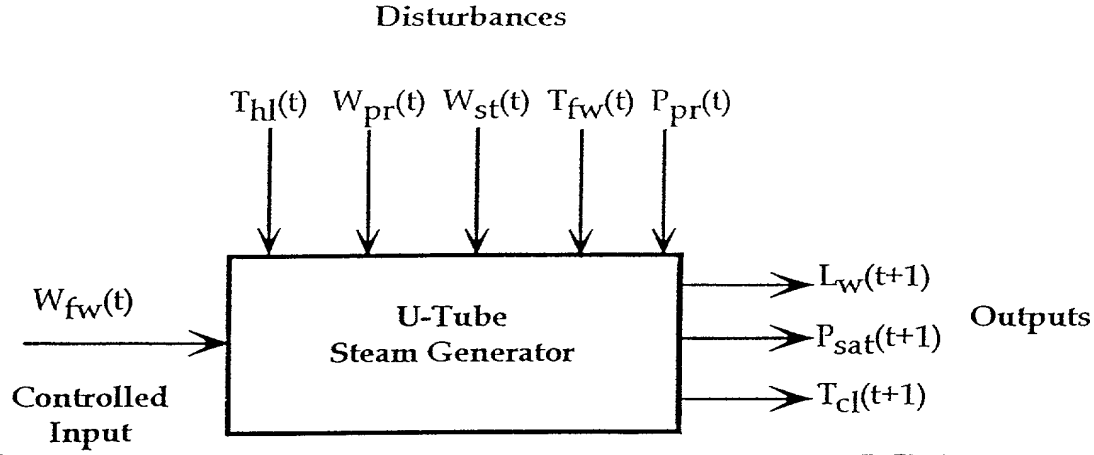


Figure 77. Block Diagram of the UTSG Process Simulator.

The UTSG has one control input which is the feedwater flow rate, $W_{fw}(t)$, and five disturbance inputs, namely, the hot-leg temperature, $T_{hl}(t)$, the primary flow rate, $W_{pr}(t)$, the steam flow rate, $W_{st}(t)$, the feedwater temperature, $T_{fw}(t)$, and the primary pressure $P_{pr}(t)$. The measured outputs of the UTSG are three, namely, the water level, $L_w(t)$, the secondary pressure, $P_s(t)$, and the cold-leg temperature, $T_{cl}(t)$. Figure 77 shows a block diagram of the UTSG with the input, the disturbances, and the outputs denoted in the discrete-time form.

The adopted UTSG simulator has three control volumes (regions) on the primary side and four control volumes on the secondary side. The primary side region consists of the inlet plenum, the fluid volume within the tubes of the tube bundle, and the outlet plenum. The four secondary side regions are: the tube bundle region; the riser region; and the steam dome-downcomer, which is divided into a saturated volume and a subcooled volume. The saturated and subcooled volumes have a movable interface, the position of which is an unknown variable.

For the primary side model, a set of three differential equations with three unknowns is used. In matrix form these are:

$$E_1 \left(T(\tau) \right) \dot{T}(\tau) = g \left(T(\tau), Q_B(\tau), T_{hl}(\tau), P_{pr}(\tau), W_{pr}(\tau) \right), \quad (283)$$

where

$$\mathbf{E}_1(\mathbf{T}(\tau)) = \begin{bmatrix} E_{11}(\mathbf{T}(\tau)) & 0 & 0 \\ 0 & E_{22}(\mathbf{T}(\tau)) & 0 \\ 0 & 0 & E_{33}(\mathbf{T}(\tau)) \end{bmatrix}, \quad (284)$$

$$\mathbf{T}(\tau) = [T_1(\tau), T_2(\tau), T_3(\tau)]^T, \quad (285)$$

and where $\mathbf{g}(\cdot)$ is a three dimensional vector forcing function, $T_1(\tau)$, $T_2(\tau)$ and $T_3(\tau)$ are the temperatures of the three primary side control volumes, and $Q_B(\tau)$, the heat load, is the thermal heat transferred from the primary side to the secondary side across the tube bundle region. The heat load is calculated using the following expression:

$$Q_B(\tau) = U_{\text{over}} A_o \Delta T_{LM}(\tau), \quad (286)$$

where U_{over} is the overall heat transfer coefficient, A_o is the total outside surface area of the tubes, $\Delta T_{LM}(\tau)$ is the log-mean temperature difference given by the following expression [78]:

$$\Delta T_{LM}(\tau) = \frac{T_1(\tau) - T_2(\tau)}{\ln \left[\frac{T_1(\tau) - T_{sat}}{T_2(\tau) - T_{sat}} \right]}, \quad (287)$$

where T_{sat} is the saturation temperature of water at P_{pr} .

For the secondary side, the mass and energy conservation equations are summed up, and the momentum equation is used to describe the recirculation flow. The secondary equations can then be represented as follows:

$$\mathbf{E}_2(\mathbf{x}_s(\tau)) \dot{\mathbf{x}}_s(\tau) = \mathbf{f}_s(\mathbf{x}_s(\tau), Q_B(\tau), W_{fw}(\tau), W_{st}(\tau), T_{fw}(\tau), \mathbf{v}(\tau)), \quad (288)$$

or,

$$\mathbf{E}_2(\mathbf{x}_s(\tau)) \frac{d}{d\tau} \begin{bmatrix} U_o(\tau) \\ V_v(\tau) \\ \alpha_N(\tau) \\ \alpha_R(\tau) \\ P_s(\tau) \\ \bar{W}(\tau) \end{bmatrix} = \begin{bmatrix} f_1(\tau) + Q_B(\tau) \\ f_2(\tau) \\ f_3(\tau) \\ f_4(\tau) \\ f_5(\tau) \\ f_6(\tau) \end{bmatrix}, \quad (289)$$

where the secondary states, $\mathbf{x}_s(\tau)$, are the internal energy at the downcomer exit, $U_o(\tau)$, the vapor volume in the steam dome, $V_v(\tau)$, the void fractions at the riser inlet and outlet, $\alpha_N(\tau)$ and $\alpha_R(\tau)$, respectively, the secondary side steam pressure, $P_s(\tau)$, and the recirculation flow rate, $\bar{W}(\tau)$. $\mathbf{v}(\tau)$ is the process noise. The right-hand side

of equation (288) represents the forcing functions and is coupled to the primary side through the heat load $Q_B(\tau)$.

The system of equations, (284) and (289), are combined to obtain the following nonlinear state space representation:

$$\mathbf{E}(\tau) \dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), u(\tau), \mathbf{w}(\tau)), \quad (290)$$

where, $\mathbf{x}(\tau)$ is the primary and secondary side combined state vector, $u(\tau)$ is the controlled input, $W_{fw}(\tau)$, $\mathbf{w}(\tau)$ is the disturbance vector and $\mathbf{f}(\cdot)$ is a nonlinear vector function.

The system of these nine nonlinear ordinary differential equations, equations (283) and (288), is solved in tandem to advance the transient simulation. The measured UTSG outputs are expressed in terms of its system states, as follows:

$$\mathbf{y}(\tau) = \mathbf{g}(\mathbf{x}(\tau)), \quad (291)$$

with,

$$\mathbf{y}(\tau) = [L_w(\tau), T_3(\tau), P_s(\tau)]^T, \quad (292)$$

where $\mathbf{g}(\cdot)$ is a three dimensional nonlinear function vector, and where $T_3(\tau)$ is assumed to be equal to the cold-leg temperature, $T_{cl}(\tau)$.

VII.3 Process Model Description

Two UTSG process models are used in the development of the UTSG state filters. The first process model is the UTSG simulator itself *without* the process and measurement noise. This model is therefore quite accurate but it is also complex and computationally intensive.

The second UTSG process model used in this study is the scheduled model which, although not as accurate as the UTSG simulator (without process and measurement noise), it is computationally less intensive and more suitable for on-line use. The UTSG scheduled model is obtained from the nonlinear UTSG model by numerically linearizing the nonlinear UTSG model about several equilibrium points

at different operating power levels. The elements of the state-space matrices that describe each linear model are fitted to a polynomial with the independent variable being a term representative of the power level at which the UTSG is operating. In this case, the independent variable was chosen to be the difference in the UTSG primary loop hot-leg and cold-leg temperatures, $\Delta T(\tau)$. This procedure of fitting the elements of the state-space matrices of the model to an independent variable is referred to as scheduling. The reader is referred to Menon [48] for further details on the linearization and scheduling procedures of the UTSG nonlinear model.

A linear UTSG model, valid for a particular power level, can be represented in a state-space form as:

$$\delta \dot{\mathbf{x}}(\tau) = \mathbf{A} \delta \mathbf{x}(\tau) + \mathbf{B} \delta u(\tau) + \mathbf{G} \delta \mathbf{w}(\tau), \quad (293)$$

$$\delta y(\tau) = \mathbf{C} \delta \mathbf{x}(\tau), \quad (294)$$

where $\delta \mathbf{x}(\tau)$, $\delta u(\tau)$ and $\delta \mathbf{w}(\tau)$ are the perturbed state, input and disturbance vectors, respectively, about the equilibrium point. Additionally, \mathbf{A} is the state matrix (9×9), \mathbf{B} is the state input vector (9×1), \mathbf{G} is the state disturbance matrix, and \mathbf{C} is the output vector (1×9).

Each of the different linear models obtained by numerical linearization corresponding to different power level. Therefore each is accurate only for a limited range of operation around the corresponding power level. To extend the use of these linear models to operating conditions/transients beyond the range of any single linear UTSG model, the elements of the state space matrices, \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{G} are fitted to polynomials of varying degrees with the independent (scheduling) variable being $\Delta T(\tau)$. The UTSG scheduled model is represented by the equations:

$$\delta \dot{\mathbf{x}}(\tau) = \mathbf{A}(\Delta T(\tau)) \delta \mathbf{x}(\tau) + \mathbf{B}(\Delta T(\tau)) \delta u(\tau) + \mathbf{G}(\Delta T(\tau)) \delta \mathbf{w}(\tau), \quad (295)$$

$$\delta y(\tau) = \mathbf{C}(\Delta T(\tau)) \delta \mathbf{x}(\tau), \quad (296)$$

where $\delta \mathbf{x}(\tau)$, $\delta u(\tau)$ and $\delta \mathbf{w}(\tau)$ are the perturbed state, input and disturbance vectors, respectively. $\mathbf{A}(\cdot)$, $\mathbf{B}(\cdot)$, $\mathbf{G}(\cdot)$ and $\mathbf{C}(\cdot)$ are the scheduled UTSG model system matrices with elements that are functions of $\Delta T(\tau)$.

The scheduled UTSG model obtained is simpler and faster in execution than the nonlinear UTSG model. The accuracy of this model is only moderate because of the inaccuracies introduced by the linearization process. However, it serves well the purpose of testing the state filtering methods developed in the earlier chapters. The predictor equations, representing either the UTSG simulator model or the scheduled model in discrete-time form, can be expressed by the following equations:

$$\hat{\mathbf{x}}(t+1|t) = \mathbf{f}_m(\hat{\mathbf{x}}(t|t), u(t), \mathbf{w}(t)), \quad (297)$$

$$\hat{\mathbf{y}}(t+1|t) = \mathbf{h}_m(\hat{\mathbf{x}}(t+1|t)), \quad (298)$$

where, $\hat{\mathbf{x}}(t+1|t)$ is a vector having the same significance as the nine UTSG process model states, $\hat{\mathbf{y}}(t+1|t)$ represents the UTSG process model outputs, and $\mathbf{f}_m(\cdot)$ and $\mathbf{h}_m(\cdot)$ are nonlinear functions.

VII.4 Neural Network State Filters

In this section, four state filters, Filter 1, Filter 2, Filter 3, and Filter 4, based on the hybrid adaptive and adaptive state filters discussed in Chapter IV, are developed and investigated. Filters 1, 2, and 3 are developed to estimate the riser void fraction, $\alpha_R(t)$. Filter 4 is developed to estimate the primary heat transfer coefficient (HTC), U_{over} , which is a constant parameter.

Filter 1 is a hybrid adaptive state filter which uses the scheduled UTSG process model as the model predictor. Filter 2 is also a hybrid adaptive state filter, like Filter 1, which uses the UTSG simulator process model without process and measurement noise. Therefore, Filter 2 is developed with a more accurate UTSG process model than Filter 1. Filters 3 and 4 are adaptive NN state filters and are developed using the UTSG simulator process model.

The estimation and validation data sets used in the development and evaluation of the NN models involved in the state filters are a collection of UTSG simulator process data corresponding to step and ramp changes in operating power level. These

Table 8. UTSG Process System Estimation and Validation Data Set Used by the Hybrid Adaptive and Adaptive NN Riser Void Fraction Filters 1,2 and 3.

Estimation Data Set	
Weight Update	Load Changes (% of Full Power)
Yes	5% to 8% (Step)
	10% to 7% (Step)
	15% to 20% (Step)
	10% to 15% (Step)
	5% to 15% (Ramp) (0.8%/minute)
	20% to 10% (Ramp) (1%/minute)
	10% to 5% (Ramp) (0.5%/minute)
No	20% to 15% (Step)
	10% to 20% (Ramp) (1%/minute)
	10% to 13% (Step)
Validation Data Set	
No	5% to 10% (Ramp) (0.6%/minute)
	15% to 18% (Step)

operating power level changes are effected by changes in the UTSG inputs: the Hot-Leg temperature, $T_{hl}(t)$; the Steam Flow rate, $W_{st}(t)$; the Feedwater Temperature, $T_{fw}(t)$; and the controlled input, the Feedwater Flow rate, $W_{fw}(t)$. These disturbances and the controlled input shall henceforth be referred to, collectively, by the vector $u(t)$. In Filters 1, 2 and 3, the only output measurement used is the UTSG Water Level, $L_w(t)$. In Filter 4, all three output measurements of the UTSG process simulator, the Water Level, $L_w(t)$, Secondary Pressure, $P_s(t)$, and the Cold-Leg Temperature, $T_{cl}(t)$, are used. The estimation and validation data sets used in the design and validation of Filters 1, 2 and 3 are listed in Table 8. The estimation and

Table 9. UTSG Process System Estimation and Validation Data Sets Used by the Adaptive NN HTC Filter 4.

Estimation Data Set		
Weight Update	U_{over} (% of Nominal Value)	Load Changes (% of Full Power)
Yes	100%, 70%, 40%	5% to 8% (Step)
		10% to 7% (Step)
		15% to 20% (Step)
		10% to 15% (Step)
		5% to 10% (Ramp) (0.8%/minute)
		20% to 15% (Ramp) (1%/minute)
No	60%	20% to 15% (Step)
	50%	10% to 13% (Step)
	100%	10% to 15% (Ramp) (1%/minute)
Validation Data Set		
No	70%	5% (Steady-State)
	60%	10% (Steady-State)
	50%	15% (Steady-State)
	70%	20% (Steady-State)
	65%	15% to 20% (Ramp) (1%/minute)
	45%	15% to 18% (Step)
	55%	15% to 20% (Ramp) (1%/minute)
	35%	10% to 13% (Step)

validation data sets used in the development and evaluation of the NN models in Filter 4 are listed in Table 9. All simulations of the UTSG process are done in the low power operation range which corresponds to less than 20% of full operational power. This is because the low power operation regime reveals the greatest nonlinearities in the UTSG process. In addition, process and measurement noise, which is zero-mean, white, Gaussian with 0.05 sd, is added to the collected data.

The following subsections discuss in detail, the design and evaluation of the NN state filters as applied to the UTSG process system. All simulations of the NNs were performed on a DEC AlphaServerTM 4/275 computer.

VII.4.1 Filter 1 - Hybrid Adaptive Filter for the Riser Void Fraction

Filter 1 is a hybrid adaptive NN state filter used to filter the riser void fraction, $\alpha_R(t)$, in the UTSG process system. In this filter, the scheduled UTSG model, described in the earlier subsections, is used as the process model (predictor). The estimation and validation data set transients is depicted in Table 8. The total number of samples in the estimation data set is 2800. Of the 2800 samples in the estimation data set, only 2050 samples are used in updating the weights of the NNs, while the remaining 750 samples are used to evaluate the generalization of the NN model. The number of samples in the validation data set is 500 and comprises two transients, a ramp and step operating power change, respectively. The sampling time for the estimation and validation data set is 4 seconds ($\Delta t = 4s$).

Filter 1 uses two NNs: the NN error model and the NN state filter. The design of these NNs is discussed in the following subsections.

VII.4.1.1 NN Error Model Development

The NN error model is first chosen to be a three-layer FMLP network with 6 nodes in the first layer (corresponding to the 6 inputs: $u(t)$, $L_w(t+1)$, and $\hat{L}_{ew}(t|t)$). The signal $L_{ew}(t)$ is the difference between the UTSG process simulator measurements, $L_w(t)$, and the model predictor (in this case, the scheduled UTSG process model) output, $\hat{L}_w(t|t-1)$. In the output layer, there is only one node. This node corresponds to the single output $\hat{L}_{ew}(t+1|t+1)$. The number of nodes in the second (hidden) layer is initially set to 4. The NN error model is then trained with the estimation data set. The target for the NN error model output is $L_{ew}(t+1)$ which is computed from the estimation data set.

Training is stopped when the NMSE of the estimation data set (no weight update part) just begins to increase (indicating the start of over-training). The NMSE

is noted, the number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. The NN architecture with the lowest NMSE in the estimation data set (no weight update) is chosen. In this case, the chosen NN architecture was 6-5-1 with a NMSE of 1.98%. The RMLP network was also used to develop the NN error model. However, the lowest NMSE obtained with a RMLP network was about 62%. Therefore, the FMLP 6-5-1 network was chosen as the NN error model because of the much greater NMSE of the RMLP networks.

VII.4.1.2 NN State Filter Development

The NN state filter architecture is first chosen to be a three-layer FMLP network with 3 nodes in the first layer, corresponding to 3 inputs, and 1 node in the third layer, corresponding to 1 output. The 3 inputs to the first layer are the water level, $L_w(t+1)$, the residual $\epsilon(t+1)$ where $\epsilon(t+1) = L_w(t+1) - \hat{L}_{cw}(t+1|t+1)$, and the model riser void fraction $\alpha_{mR}(t)$. The signal $\hat{L}_{cw}(t+1|t)$ is the *corrected* process model predictor output and is the sum of the model predictor output, $\hat{L}_w(t+1|t)$, and the output of the NN error model, $\hat{L}_{ew}(t+1|t+1)$. The single output of the NN state filter is $\hat{\alpha}_{NN,R}(t+1|t+1)$. Initially, the number of nodes in the hidden layer is chosen to be 3. The NN state filter is then trained with the estimation data set. The target is the state $\alpha_{mR}(t+1)$ as provided by the UTSG scheduled process model.

Training is stopped when the NMSE of the estimation data set (no weight update) just begins to increase (indicating the start of over-training). The NMSE is noted and the number of nodes in the hidden layer is increased and the training cycle is repeated. It was determined that the FMLP NN with an architecture of 3-4-1 was the best network. The NMSE for the estimation data set (no weight update) for this network is 0.11%. A block diagram of the setup in Filter 1 is shown in

Figure 78. A comparison between the UTSG process simulator outputs, $L_w(t)$, and the corrected scheduled UTSG process model (predictor) output, $\hat{L}_{cw}(t|t)$, is shown in Figure 79. The comparison between the UTSG process simulator state values, $\alpha_R(t)$, the scheduled UTSG process model state values, $\alpha_{mR}(t)$, and the FMLP NN state filter outputs, $\hat{\alpha}_{NN,R}(t|t)$ is shown in Figure 80.

The RMLP was not used to develop the NN state filter since the FMLP performed adequately and the additional complexity of the RMLP did not warrant its use.

VII.4.2 Filter 2 - Hybrid Adaptive Filter for the Riser Void Fraction

The method used to develop the riser void fraction hybrid adaptive state filter is as described in the previous section. However, in this scheme, a more accurate UTSG process model predictor than the scheduled UTSG process model is used. The UTSG process model used in Filter 2 is the UTSG simulator itself without the process and measurement noise. This UTSG process model is also referred to as the simulator model.

For the development of Filter 2, process data for the estimation and validation data set is collected every second ($\Delta t = 1s$). The estimation and validation data set is depicted in Table 8. The total number of samples in the estimation data set is 11200. Out of the 11200 samples in the estimation data set, only 8200 samples are used in updating the weights of the NNs and the remaining 3000 samples are used to evaluate the NN model training (no weight update). The number of samples in the validation data set is 2000.

Filter 2 uses two NNs: the NN error model and the NN state filter. The design of these NNs is discussed in the following subsections.

VII.4.2.1 NN Error Model Development

The NN error model is first chosen to be a three-layer FMLP network with 6 nodes in the first layer corresponding to the 6 inputs: $u(t)$, $L_w(t+1)$ and $\hat{L}_{cw}(t|t)$. The

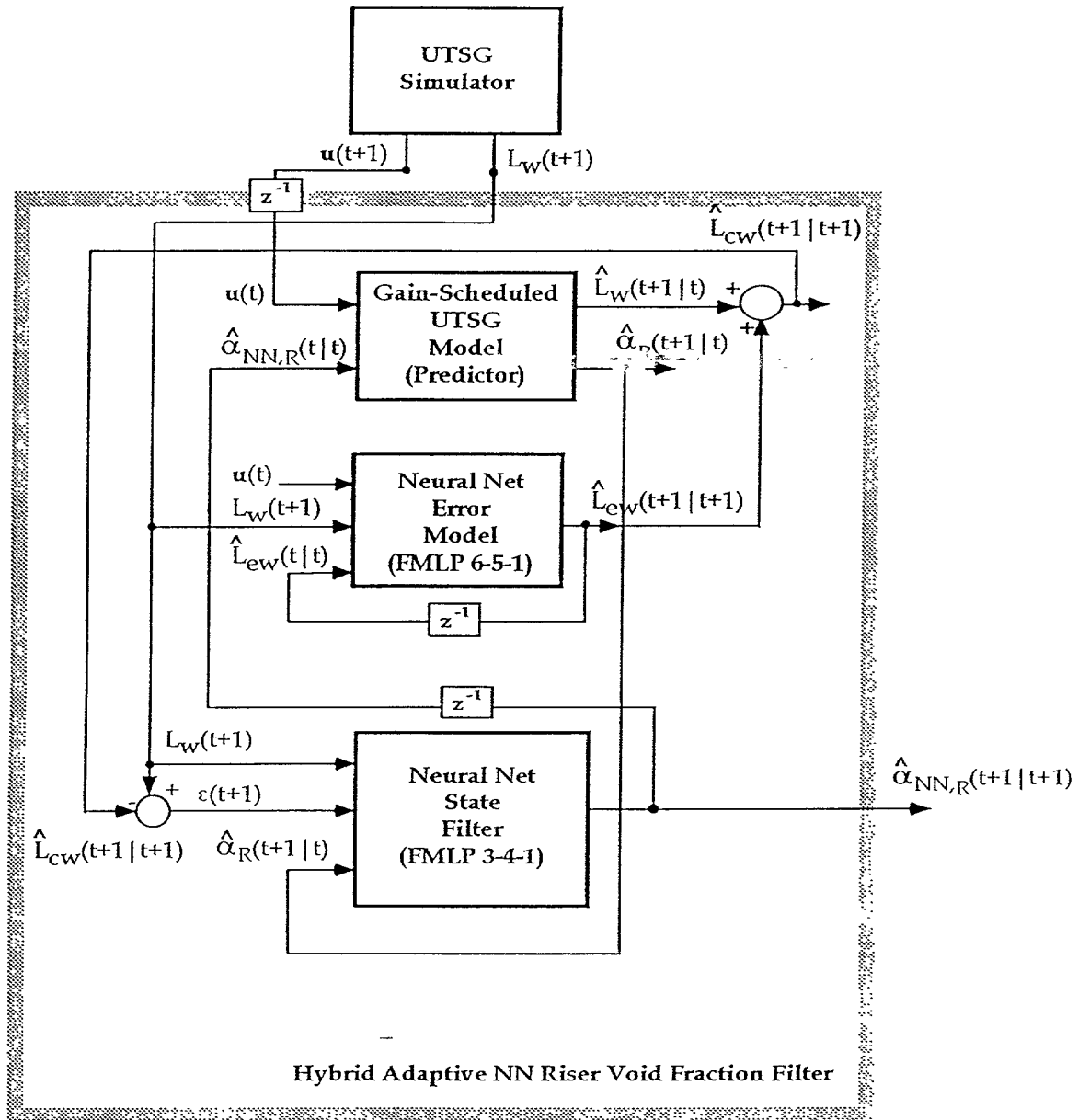


Figure 78. Block Diagram of the UTSG Process Hybrid Adaptive NN Riser Void Fraction Filter 1.

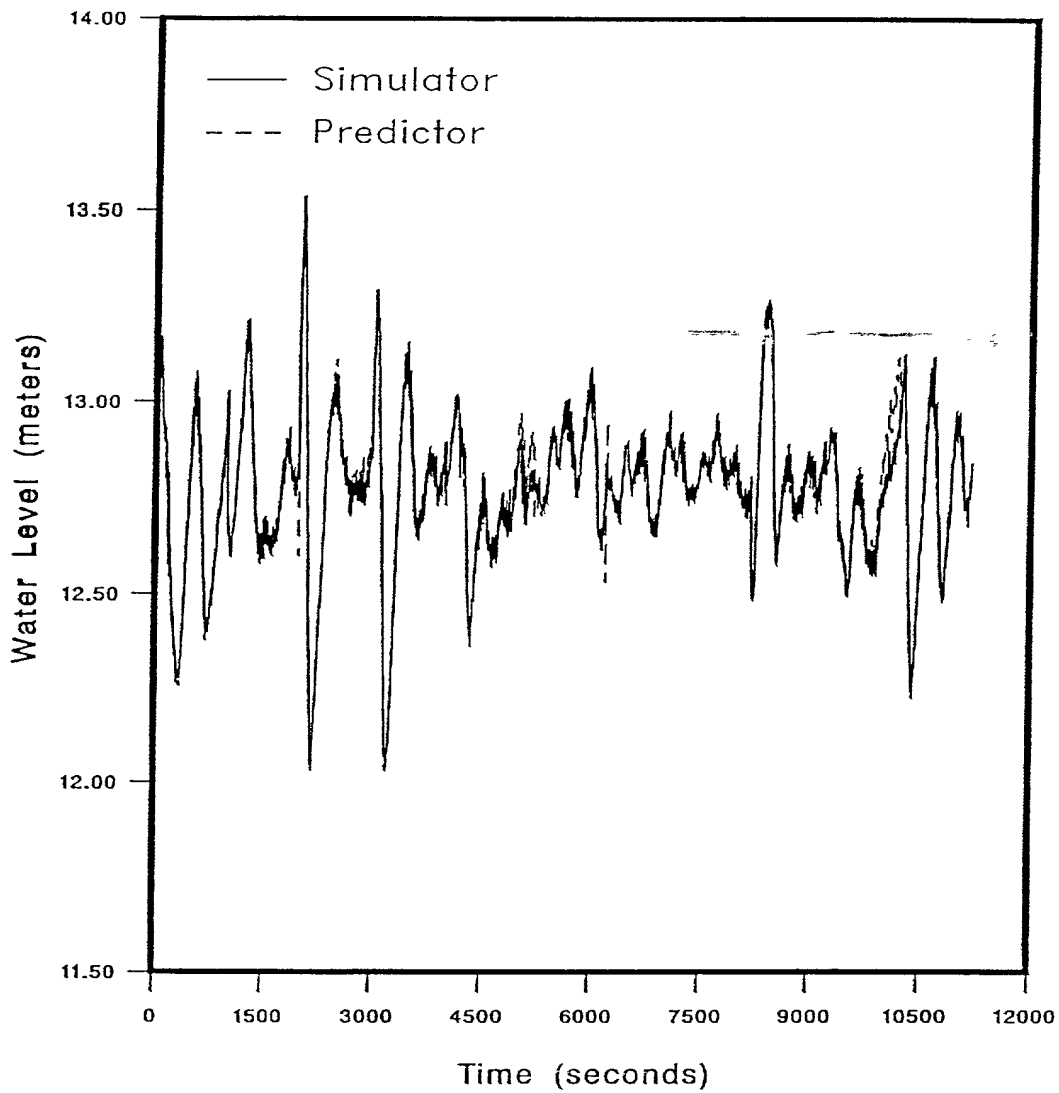


Figure 79. UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using the Estimation Data Set as Inputs.

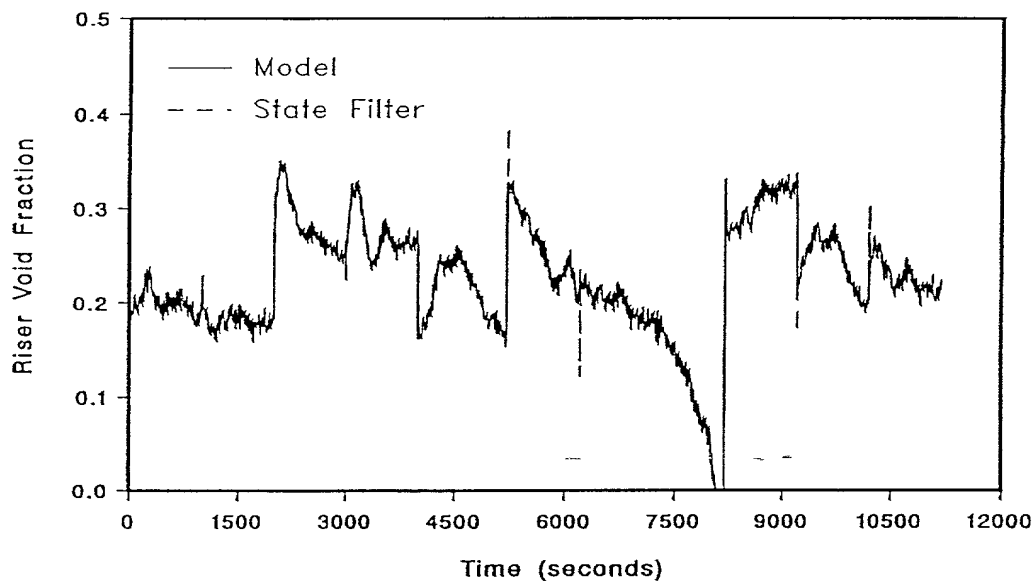
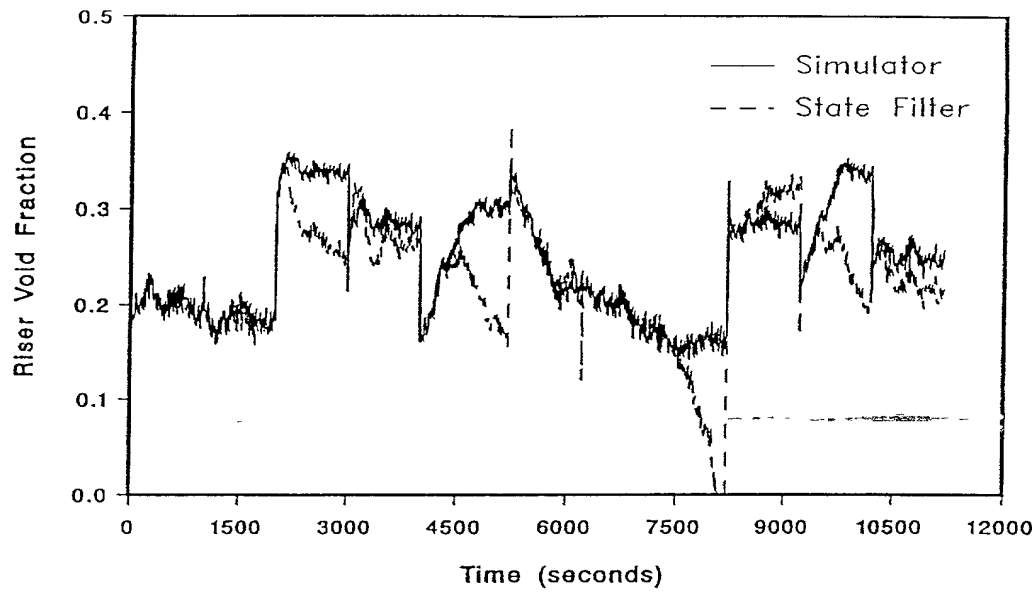


Figure 80. UTSG Process Riser Void Fraction Response, from the Simulator, Scheduled Model, and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using the Estimation Data Set as Inputs.

signal $L_{ew}(t)$ is the difference between the process simulator measurements, $L_w(t)$, and the model predictor output, $\hat{L}_w(t|t-1)$. There is only one node in the output layer. This node corresponds to the single output $\hat{L}_{ew}(t+1|t+1)$. The number of nodes in the second (hidden) layer is set to 4 initially. The NN error model is then trained with the estimation data set. The target is $L_{ew}(t+1)$ which is determined from the estimation data set.

Training is stopped when the NMSE of the estimation data set (no weight update part) just begins to increase (indicating the start of over-training). The NMSE is noted, the number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. The NN architecture with the lowest NMSE in the estimation data set (no weight update) is chosen. In this case, the chosen NN architecture was 6-5-1 with a NMSE of 0.73%. The RMLP network was also used to develop the NN error model. However, the lowest NMSE obtained with a RMLP network was about 45%. Therefore, the 6-5-1 FMLP network was chosen as the NN error model.

VII.4.2.2 NN State Filter Development

The NN state filter architecture is first chosen to be a three-layer FMLP network with 3 nodes in the first layer, corresponding to 3 inputs, and 1 node in the third layer, corresponding to 1 output. The 3 inputs to the first layer are the water level, $L_w(t+1)$, the residual $\epsilon(t+1)$ where $\epsilon(t+1) = L_w(t+1) - \hat{L}_{cw}(t+1|t+1)$, and the void fraction $\alpha_{mR}(t)$. The signal $\hat{L}_{cw}(t+1|t+1)$ is the *corrected* process model (predictor) output and is the sum of the model predictor output, $\hat{L}_w(t+1|t)$, and the output of the NN Error Model, $\hat{L}_{ew}(t+1|t+1)$. The single output of the NN state filter is $\hat{\alpha}_{NN,R}(t+1|t+1)$. The number of nodes in the hidden layer is initially chosen to be 3. The NN state filter is then trained with the estimation data set. The target is the state $\alpha_{mR}(t+1)$ as provided by the UTSG process model (simulator without process and measurement noise).

Training is stopped when the NMSE of the estimation data set (no weight update) just begins to increase (indicating the start of over-training). The NMSE is

noted, the number of nodes in the hidden layer is increased and the training cycle is repeated. It was determined that the FMLP NN with an architecture of 3-5-1 was the best network. The NMSE for the estimation data set (no weight update) for this network is 0.03%. A block diagram Filter 2 is shown in Figure 81. A comparison between the actual UTSG process simulator outputs, $L_w(t)$, and the corrected UTSG model predictor output, $\hat{L}_{cw}(t|t)$, is shown in Figure 82. The comparison between the UTSG process simulator state values, $\alpha_R(t)$, the UTSG simulator model state values, $\alpha_{mR}(t)$, and the NN state filter outputs, $\hat{\alpha}_{NN,R}(t|t)$, is shown in Figure 83.

The RMLP Network was not used to develop the NN state filter since the FMLP network performed adequately and the additional complexity of the RMLP network did not warrant its use.

VII.4.3 Filter 3 - Adaptive Filter for the Riser Void Fraction

Filter 3 is based on the adaptive NN state filter discussed in Chapter IV. For the development of the NNs in Filter 3, process data is collected every 5 seconds ($\Delta t = 5s$) and the estimation data set transients are shown in Table 8. The total number of samples in the estimation data set is 2240. Of the 2240 samples in the estimation data set, 1640 samples are used in updating the weights of the NNs and the remaining 600 samples are used to evaluate the NN model training. The number of samples in the validation data set is 400. The state, $\alpha_{mR}(t)$, is obtained from the UTSG Simulator model and is used to develop the appropriate NN models.

Three NN models are used in this estimation scheme: the output predictor, the state predictor and the state filter. The NN output predictor is developed first. The NN state predictor is developed with the NN output predictor running concurrently. Finally, the NN state filter is developed with both the NN output predictor and the NN state predictor running concurrently. The development of each of these models is discussed in greater detail in the following subsections.

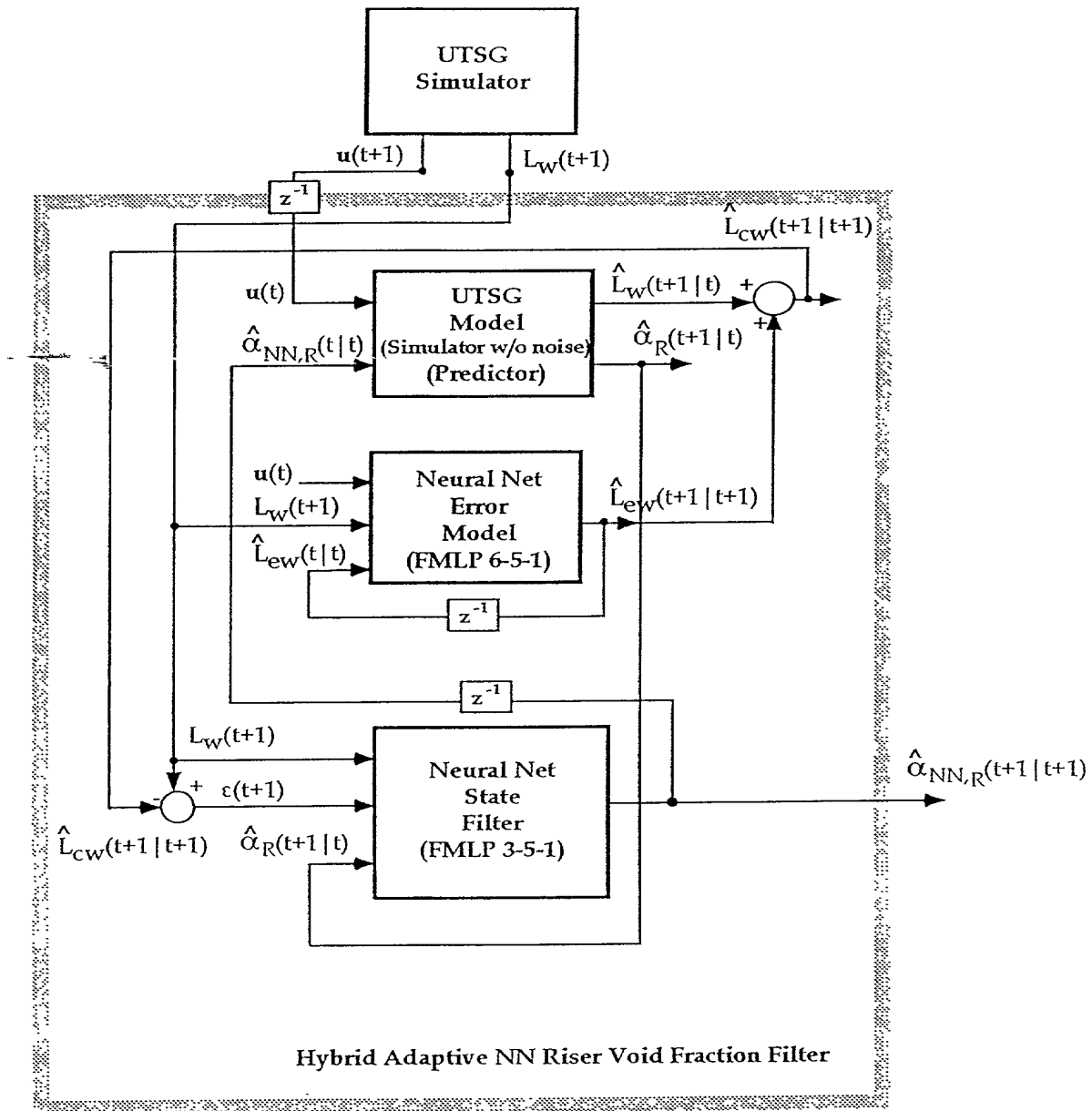


Figure 81. Block Diagram of the UTSG Process Hybrid Adaptive NN Riser Void Fraction Filter 2.

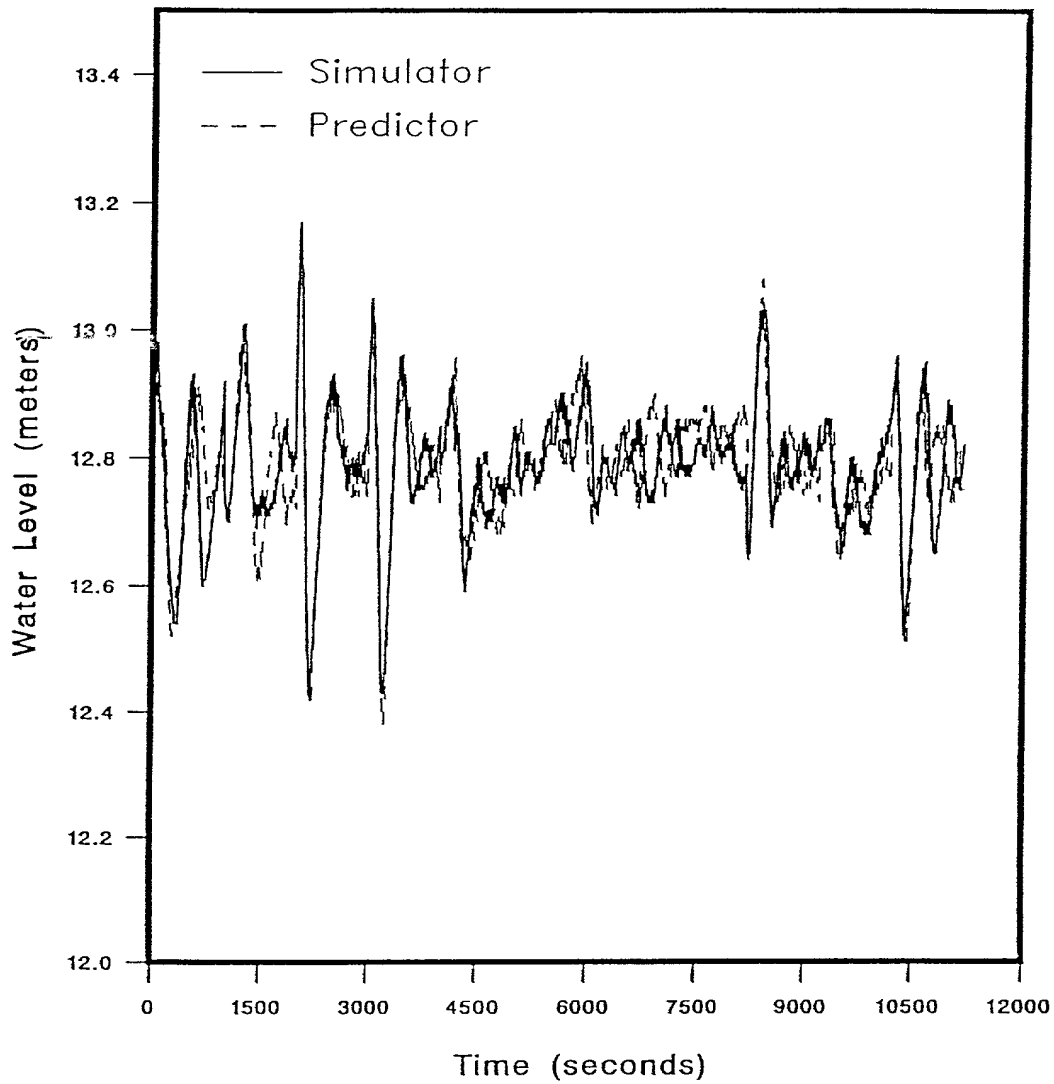


Figure 82. UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using the Estimation Data Set as Inputs.

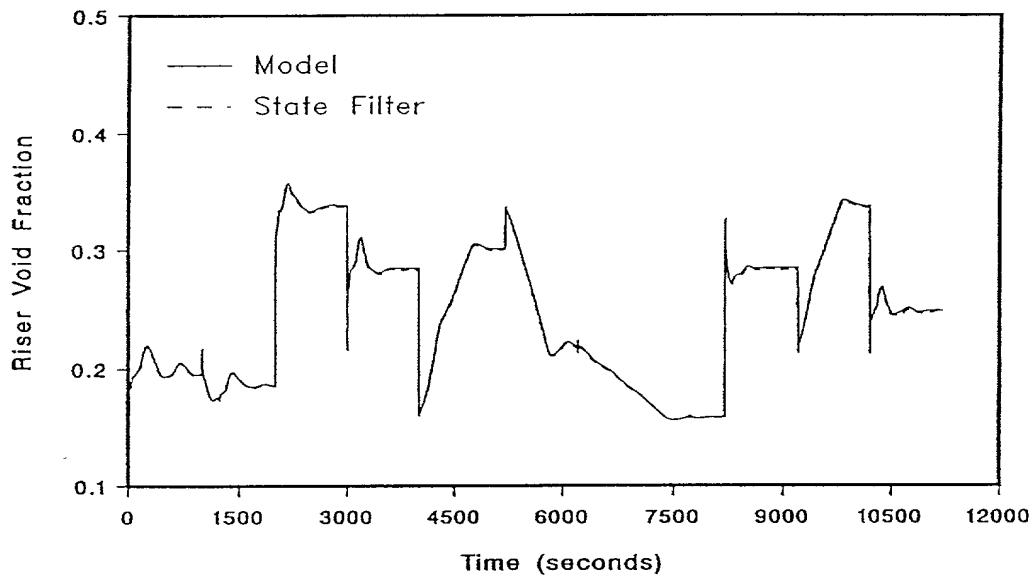
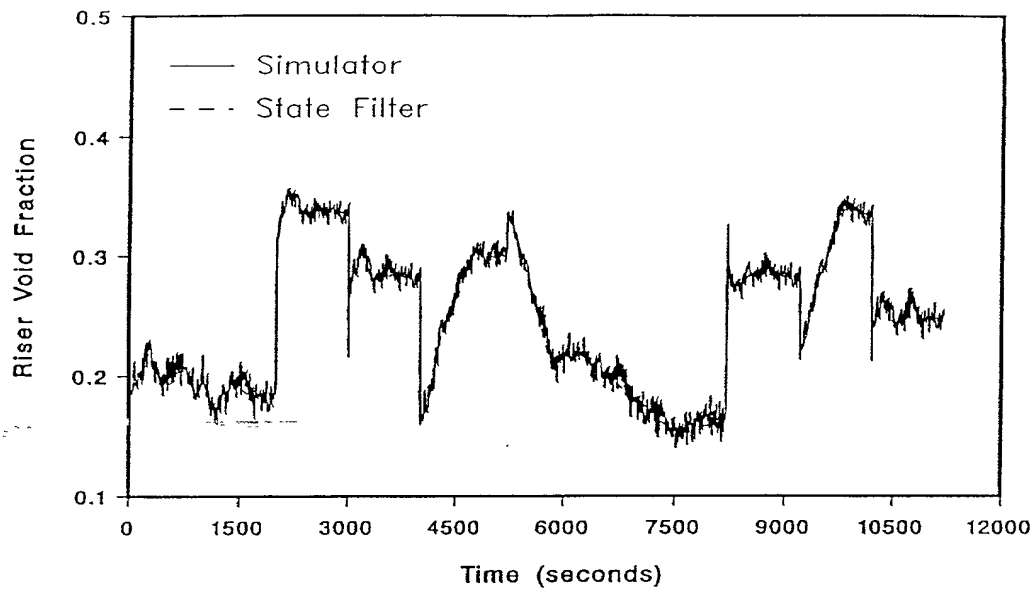


Figure 83. UTSG Process Riser Void Fraction Response, from the Simulator, Simulator Model, and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using the Estimation Data Set as Inputs.

VII.4.3.1 NN Output Predictor Development

The NN output predictor architecture is initially chosen to be a three-layer RMLP network with 6 nodes in the first layer, corresponding to 6 inputs, and one node in the third layer, corresponding to 1 output. The single output of the third layer is the SSP of the UTSG water level, $\hat{L}_{NN,w}(t+1|t)$. The 6 inputs to the first layer are the process inputs, $u(t)$, the state $\alpha_{mR}(t)$ and the delayed output of the NN Model itself, $\hat{L}_{NN,w}(t|t-1)$. Initially, the number of nodes in the hidden layer is set to 3. The NN output predictor is then trained with the estimation data set. The target is the UTSG simulator model output, $L_{mw}(t+1)$.

The training of the NN output predictor is done until the stopping criterion is reached; that is, until the NMSE of the estimation data set (no weight update) just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A suitable number of NNs are developed in this way and the best among them is chosen. In this case, the best NN output predictor architecture was found to be 6-3-1. The NMSE of the output, in this case, is 0.4%. A comparison between the UTSG water level and the NN output predictor response for the estimation data set is shown in Figure 84.

VII.4.3.2 NN State Predictor Development

The NN output predictor weights are now fixed and no further training will be done on this model. The NN state predictor is chosen to be a three-layer RMLP with 6 inputs in the first layer, corresponding to 6 inputs, and 1 node in the third layer, corresponding to one output. The output of the third layer is the SSP of the state, $\hat{\alpha}_{NN,R}(t+1|t)$. The inputs to the first layer are: the output from the NN output predictor, $\hat{L}_{NN,w}(t|t-1)$, the inputs to the process system, $u(t)$, and the delayed output of the NN state predictor, $\hat{\alpha}_{NN,R}(t|t-1)$. The number of nodes in the hidden layer is set to 3 initially. The NN state predictor is then trained with the estimation data set. The target is the state, $\alpha_{mR}(t+1)$ as provided by the UTSG simulator model.

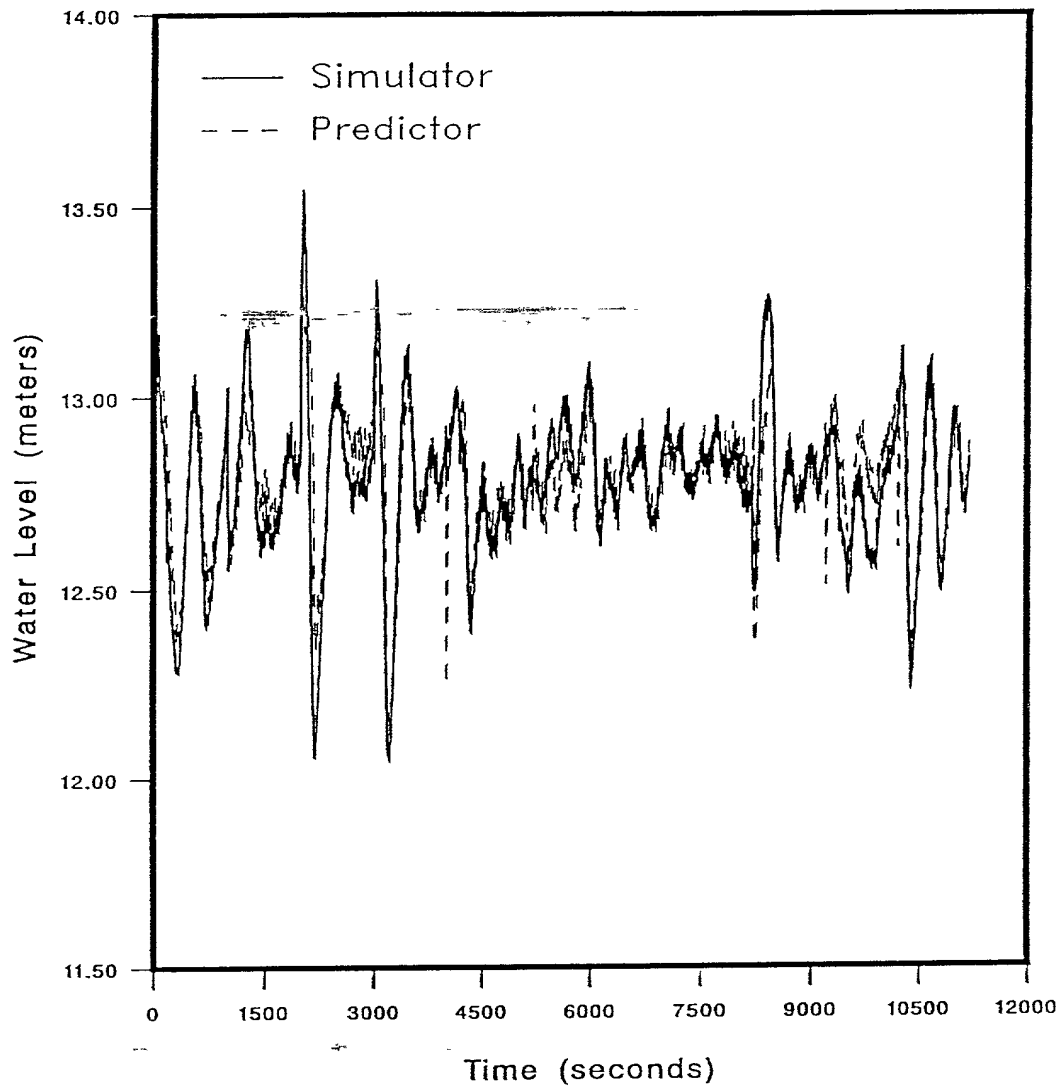


Figure 84. UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using the Estimation Data Set as Inputs.

The NN state predictor is trained until the NMSE of the estimation data set (no weight update) just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. Based on the lowest NMSE on the estimation data set (no weight update), a RMLP network with an architecture of 6-4-1 was determined to be the best. The average NMSE for this architecture was 0.143%. FMLP networks were not used to obtain the NN state predictor because the complexity of the process system necessitates the use of the RMLP networks.

VII.4.3.3 NN State Filter Development

The weights of the NN output predictor and NN state predictor are fixed and no further training is done on these networks. The NN state filter architecture is first chosen to be a three-layer FMLP NN with 3 nodes in the input (first) layer, corresponding to 7 inputs, and 1 node in the third layer, corresponding to one output. The output of the third layer is the filtered state value, $\hat{\alpha}_{NN,R}(t+1|t+1)$. The 3 inputs to the first layer are the UTSG process simulator outputs, $L_w(t+1)$, the residual, $\epsilon(t+1)$ where $\epsilon(t+1) = L_w(t+1) - \hat{L}_{NN,w}(t+1|t)$, and the output from the NN state predictor, $\hat{\alpha}_{NN,R}(t+1|t)$. The target in training is the same as in the training of the NN state predictor, namely the state $\alpha_{mR}(t+1)$ obtained from the UTSG simulator model.

The initial number of nodes in the hidden layer of the NN state filter is set to 4 and training is done as described in the previous section. Different NN models are developed (with increasing number of nodes in the hidden layer). The best FMLP NN architecture was found to be a 3-5-1 network with the average NMSE in the estimation data set (no weight update) to be 0.13%.

The final setup for the scheme is shown in the block diagram in Figure 85. The performance of the NN state filter on the estimation data set is shown in Figure 86.

VII.4.4 Filter 4 - Adaptive Filter for the Primary Heat Transfer Coefficient

Filter 4 is based on the adaptive state filter discussed in Chapter IV. The objective of Filter 4 is to estimate the primary heat transfer coefficient (HTC), U_{over} , and not the riser void fraction, $\alpha_R(t)$, as in the NN filters developed in earlier sections. The qualitative difference between U_{over} and $\alpha_R(t)$ is that U_{over} is a very slowly changing parameter of the UTSG process system and can be considered to have a *constant* value for the time duration of normal operation transients of the UTSG. The riser void fraction, $\alpha_R(t)$, however, is a proper state of the UTSG system and is *expected* to change values during operating power levels changes of the UTSG. The HTC, however, can be considered as a “state” with no dynamics for the purpose of applying the adaptive state filtering methods discussed in Chapter IV.

It is apparent that the estimation data set used to develop the NNs should cover a wide range of transients of the UTSG process system with *different* U_{over} values. In addition, these variations in U_{over} should be *observable* in the UTSG process simulator output measurements. It was found, based on trial runs, that changes in U_{over} are *weakly* observable through the UTSG process simulator outputs, $L_w(t)$, $P_s(t)$ and $T_{cl}(t)$. Therefore, in order to infer variations in U_{over} with high accuracy, all three UTSG process simulator outputs were used in Filter 4. Further, it was found that the steam flow rate, W_{st} , greatly obscured the “observability” of U_{over} through the UTSG process outputs. For this reason, the estimation and validation data set are designed with the steam flow rate, $W_{st}(t)$, kept constant during power level changes. This is clearly unrealistic; however, the intention in this study is to determine the feasibility of determining the “constant” parameter U_{over} using the NN adaptive state filters discussed in earlier chapters.

The estimation and validation data sets used in the development and evaluation of the NNs in the Filter 4, as mentioned earlier, is given in Table 9. This table lists the

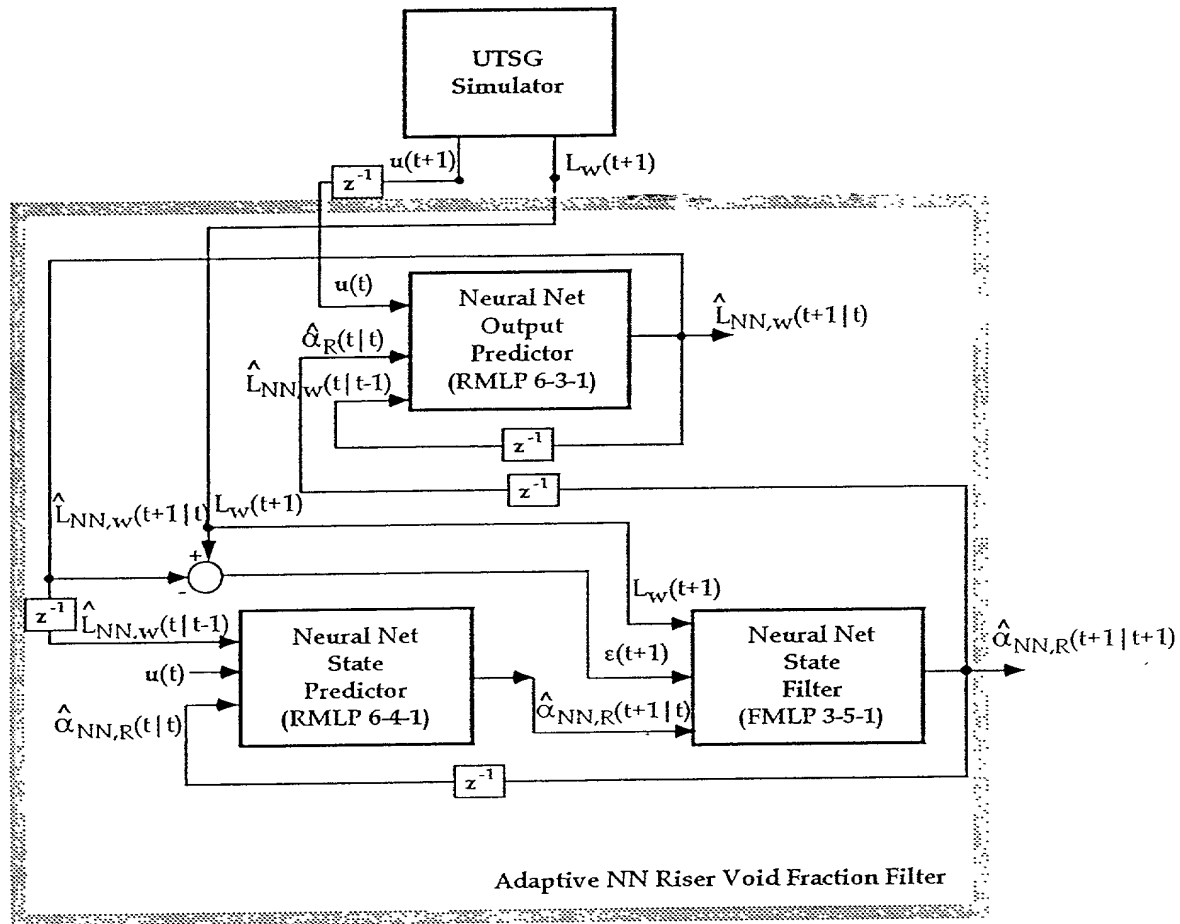


Figure 85. Block Diagram of the UTSG Process Adaptive NN Riser Void Fraction Filter 3.

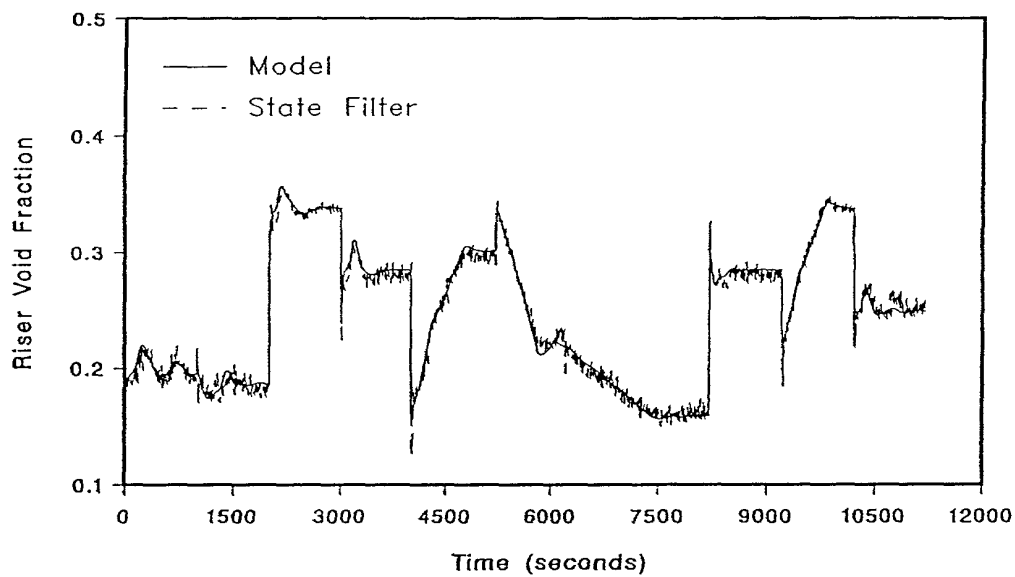
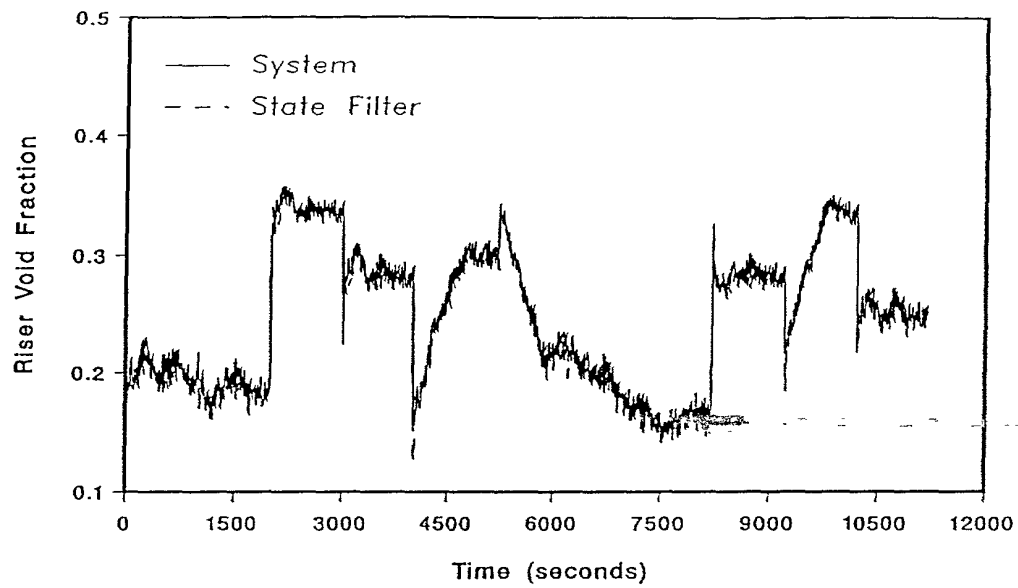


Figure 86. UTSG Process Riser Void Fraction Response, from the Simulator, Simulator Model, and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using the Estimation Data Set as Inputs.

different power level changes and steady-state operating conditions for various U_{over} values. The U_{over} values are represented in Table 9 as percentages of the nominal value ($U_{\text{over}}^{\text{nominal}} = 8874 \text{ W/m}^2 - \text{deg. K}$). Each transient listed in the table is run for 1000 seconds and data is collected every 5 seconds. Therefore, the estimation data set consists of a total of 4200 samples. Out of these 4200 samples, the NN weights are updated for the first 3600 samples and the remaining 600 samples are used in the evaluation of the NN models developed (no weight update).

Three NN models are used in this filtering scheme: the NN output predictor, the NN state predictor, and the NN state filter. The development of these models is discussed in greater detail in the following subsections.

VII.4.4.1 NN Output Predictor Development

The NN output predictor consists of three NNs; a single NN is developed for each of the outputs of the UTSG process simulator, $L_w(t)$, $T_{cl}(t)$, and $P_s(t)$, denoted by the vector $\mathbf{y}(t)$. Each of the NNs in the NN output predictor is a three-layer RMLP network. The inputs for each of the NNs are the same and are 8 in number. They are: the 4 inputs of the process system, $\mathbf{u}(t)$, the primary heat transfer coefficient, U_{over} ; and the three NN output predictor outputs (delayed by one time step), $\hat{\mathbf{y}}_{\text{NN}}(t|t-1)$. The output of each of the NNs is $\hat{\mathbf{y}}_{\text{NN}}(t+1|t) = [\hat{L}_{\text{NN},w}(t+1|t), \hat{T}_{\text{NN},cl}(t+1|t) \text{ and } \hat{P}_{\text{NN},s}(t+1|t)]$, respectively. The number of nodes in each of the NNs is initially set to 4. The three NN models are then trained concurrently with the estimation data set. The targets for each of the NNs are $L_w(t+1)$, $T_{cl}(t+1)$, and $P_s(t+1)$, respectively, and are obtained from the UTSG process simulator. Training is stopped when the NMSE of the network outputs for the estimation data set (no weight update) just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A suitable number of NN architectures are obtained in this manner and the best among them is chosen. In this case, the best NN output predictor architectures was found be 8-4-1 for $L_w(t+1)$, and $T_{cl}(t+1)$ NNs and 8-5-1 for the $P_s(t+1)$ NN. A block diagram of the NN output predictor is shown

in Figure 87. The NMSEs of the three NNs modeling the three process outputs are: 0.67% for the $L_w(t)$ predictor, 0.00004% for the $T_{cl}(t)$ predictor, and 0.002% for the $P_s(t)$ predictor. These NMSE values are evaluated for the estimation data set (no weight update). A comparison between the UTSG process system outputs and the NN output predictor outputs for the estimation data set is shown in Figure 88.

VII.4.4.2 NN State Predictor Development

The NN output predictor weights are now fixed and no further training will be done on this model. The NN state predictor is chosen to be a three-layer RMLP with 8 inputs in the first layer, corresponding to 8 inputs, and 1 node in the third layer, corresponding to one output. The output of the third layer is the SSP of the HTC, $\hat{U}_{NN,over}(t+1|t)$. The inputs to the first layer are the outputs from the NN output predictor, $\hat{y}_{NN}(t|t-1)$, the process system inputs, $u(t)$, and the delayed output of the NN state predictor, $\hat{U}_{NN,over}(t|t-1)$. The number of nodes in the hidden layer is set to 3 initially. The NN state predictor is then trained with the estimation data set. The target is the HTC, U_{over} , values of which are obtained from the UTSG simulator model.

The NN state predictor is trained until the NMSE of the estimation data set (no weight update) just begins to increase. The number of nodes in the hidden layer is increased and the training cycle is repeated. A number of NN architectures are developed in this way. The RMLP network with the lowest error was found to be a 8-7-1 network and the average NMSE for the estimation data set (no weight update) was 12%. This error was unacceptably high and so the FMLP networks are considered next. Different FMLP networks are trained, as in the case of the RMLP networks, and the best among them is chosen. In this case, a FMLP network with architecture 8-5-1, with an average NMSE in the estimation data set (no weight update) of 0.13%, was chosen as the NN state predictor.

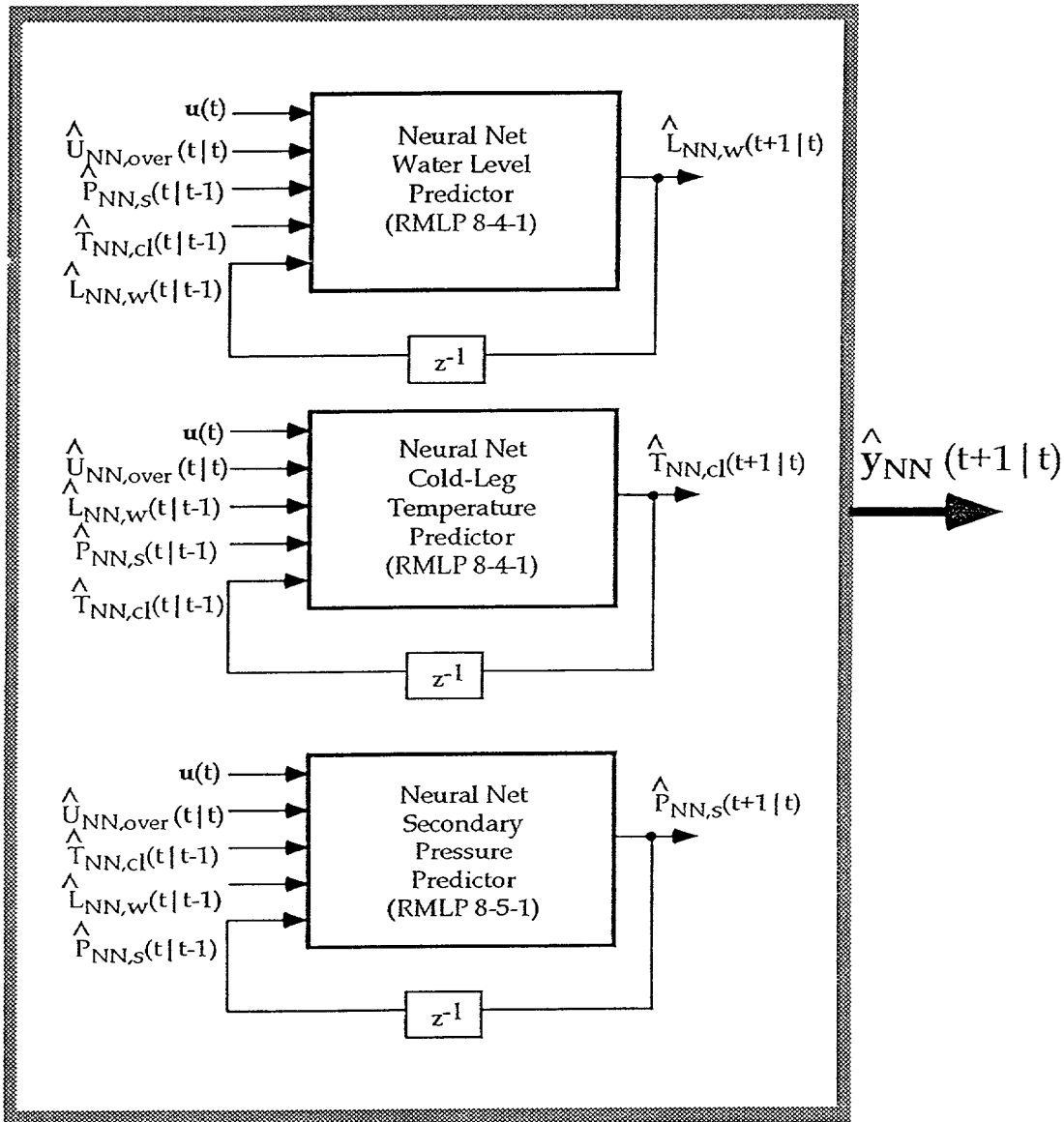


Figure 87. Block Diagram of the UTSG Process NN Output Predictor Used in the Adaptive NN HTC Filter 4.

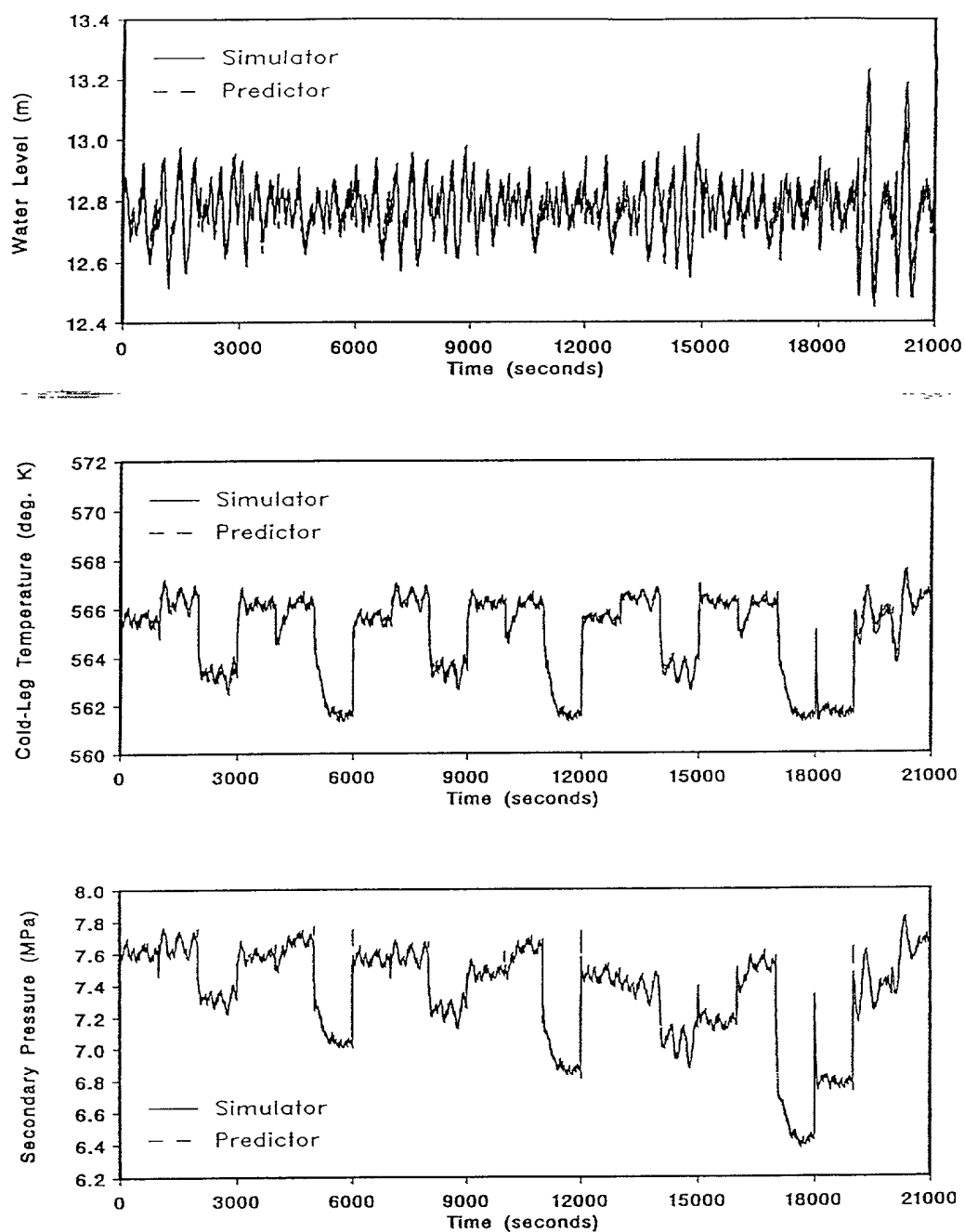


Figure 88. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using the Estimation Data Set as Inputs.

VII.4.4.3 NN State Filter Development

The weights of the NN output predictor and NN state predictor are fixed and no further training is done on these networks. The NN state filter architecture is first chosen to be a three-layer FMLP NN with 7 nodes in the input (first) layer (corresponding to 7 inputs) and 1 node in the third layer (corresponding to one output). The output of the third layer is the SSP, $\hat{U}_{\text{NN,over}}(t+1|t+1)$. The 7 inputs to the first layer are the UTSG process simulator outputs, $y(t+1)$, the residuals, $\epsilon(t+1)$ where $\epsilon(t+1) = y(t+1) - \hat{y}_{\text{NN}}(t+1|t)$, and the output from the NN state predictor, $\hat{U}_{\text{NN,over}}(t+1|t)$. The target in training is the same as in the training of the NN state predictor, namely, the process model HTC, U_{over} .

The initial number of nodes in the hidden layer is set to 4 and training is done as described in the previous section. Different NN models are developed (with increasing number of nodes in the hidden layer). The best FMLP NN architecture was found to be a 7-4-1 network with the average NMSE in the estimation data set (no weight update) to be 0.11%. The final setup for the scheme is shown in the block diagram in Figure 89. The performance of the NN state filter on the estimation data set is shown in Figure 90.

VII.5 Validation Results

The NN state filters, 1, 2, 3, and 4, developed in the previous sections, are now evaluated using the validation data set. The validation data set used in evaluating Filters 1, 2 and 3 is listed in Table 8. There are 2 individual tests in the validation data set; a ramp increase, and a step increase in the UTSG simulator operating power level.

Process and measurement noise, zero-mean, white, Gaussian with 0.05 sd, is initially added to the validation data sets. The performance of the estimation schemes is later evaluated with a higher level of process and measurement noise, zero-mean, white, Gaussian with 0.15 sd, added to the validation data sets.

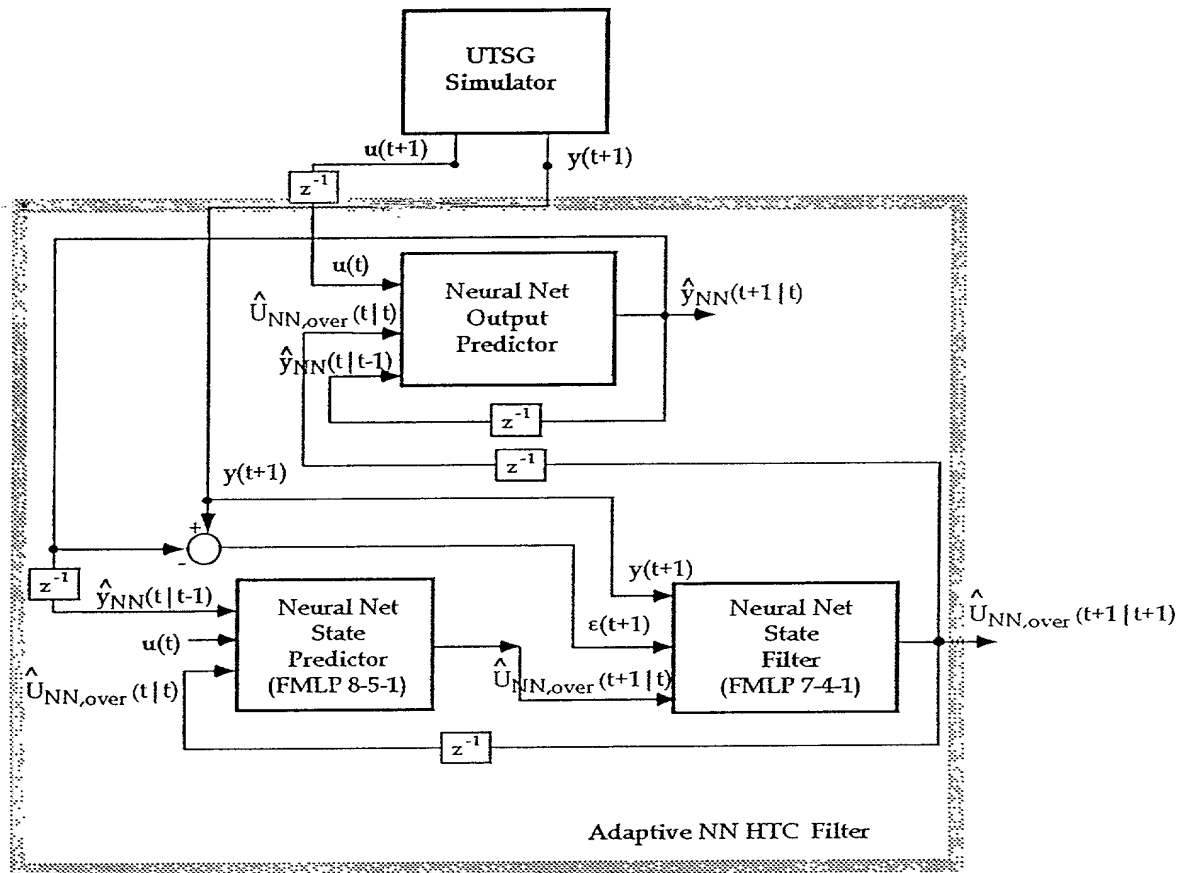


Figure 89. Block Diagram of the UTSG Process Adaptive NN HTC Filter 4.

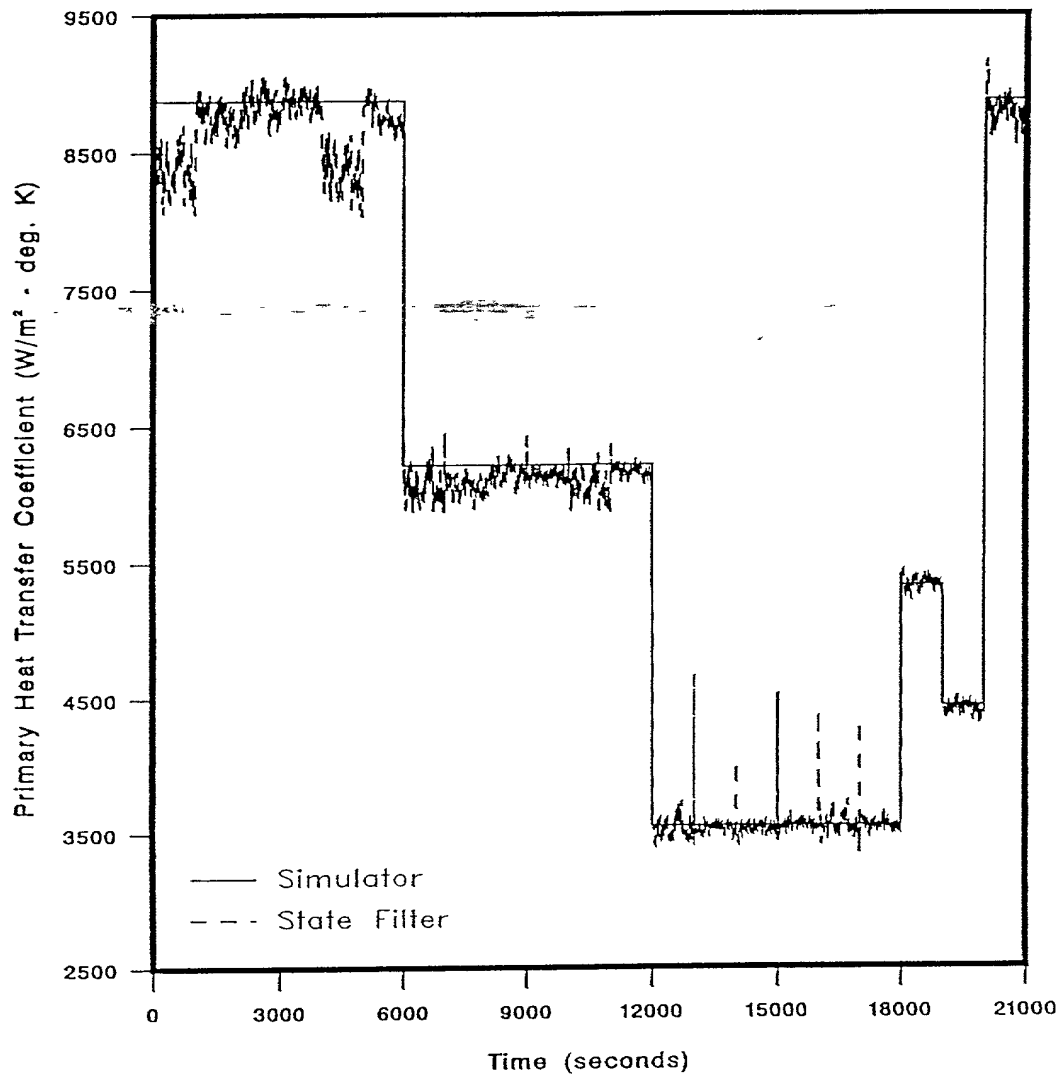


Figure 90. UTSG Process HTC, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using the Estimation Data Set as Inputs.

The validation data set used in evaluating Filter 4 is listed in Table 9. There are 8 individual tests in the validation data set; Tests 1 to 4 are conducted at steady-state power levels, Tests 5 and 7 are ramp increases in power and Tests 6 and 8 are step changes in power. These four tests are conducted with different U_{over} values and with process and measurement noise, zero-mean, white, Gaussian with 0.05 sd, added to the validation data set. The performance of the estimation scheme to Test 5 and 6 is then evaluated with a higher level of process and measurement noise, zero-mean, white, Gaussian with 0.15 sd, added to the validation data set.

The performance of the state filters, developed in the previous sections, is described in the following subsections.

VII.5.1 Filter 1 - Hybrid Adaptive Filter for the Riser Void Fraction

The performance of Filter 1 on the validation data set is now evaluated. The UTSG process scheduled model (predictor) response to the ramp input is shown in Figure 91. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter to the ramp input are shown in Figure 92. The NMSE for the state filtered values is 3.2%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 93. The average $\epsilon_{\text{Sim}}(t)$ value is 17.5% and the the average $\epsilon_{\text{Mod}}(t)$ value is 1.6%.

The UTSG process scheduled model (predictor) response to the step input is shown in Figure 94. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter are shown in Figure 95. The NMSE for the state filtered values is 1.9%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 96. The average $\epsilon_{\text{Sim}}(t)$ value is 14.2% and the the average $\epsilon_{\text{Mod}}(t)$ value is 1.1%.

Now, a higher level of process and measurement noise, zero-mean, white, Gaussian with 0.15 sd, is added to the validation data set and the performance of Filter 1 is evaluated. The UTSG process scheduled model (predictor) response to the ramp input is shown in Figure 97. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as obtained from the NN state filter to the ramp input are shown in Figure 98. The

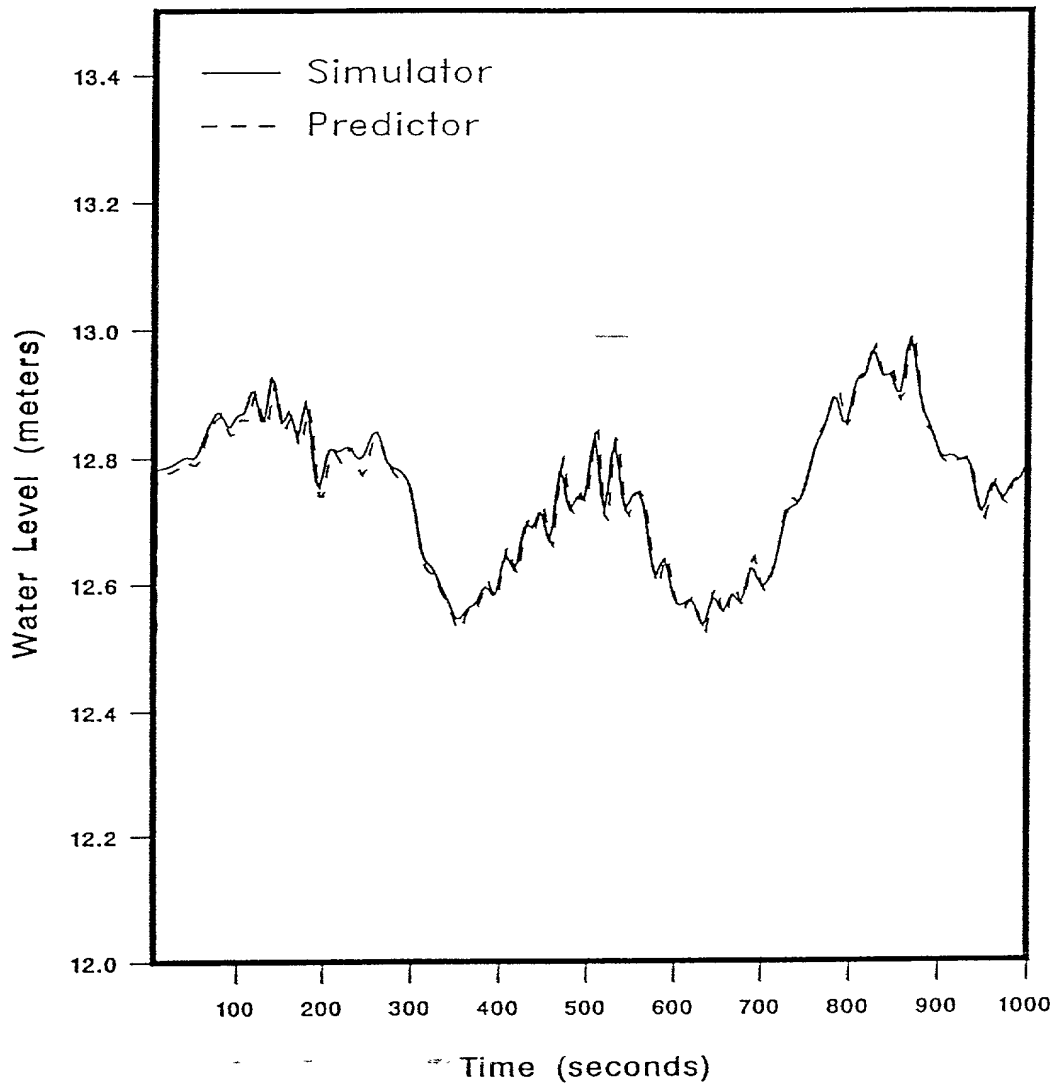


Figure 91. UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (Low Noise Environment).

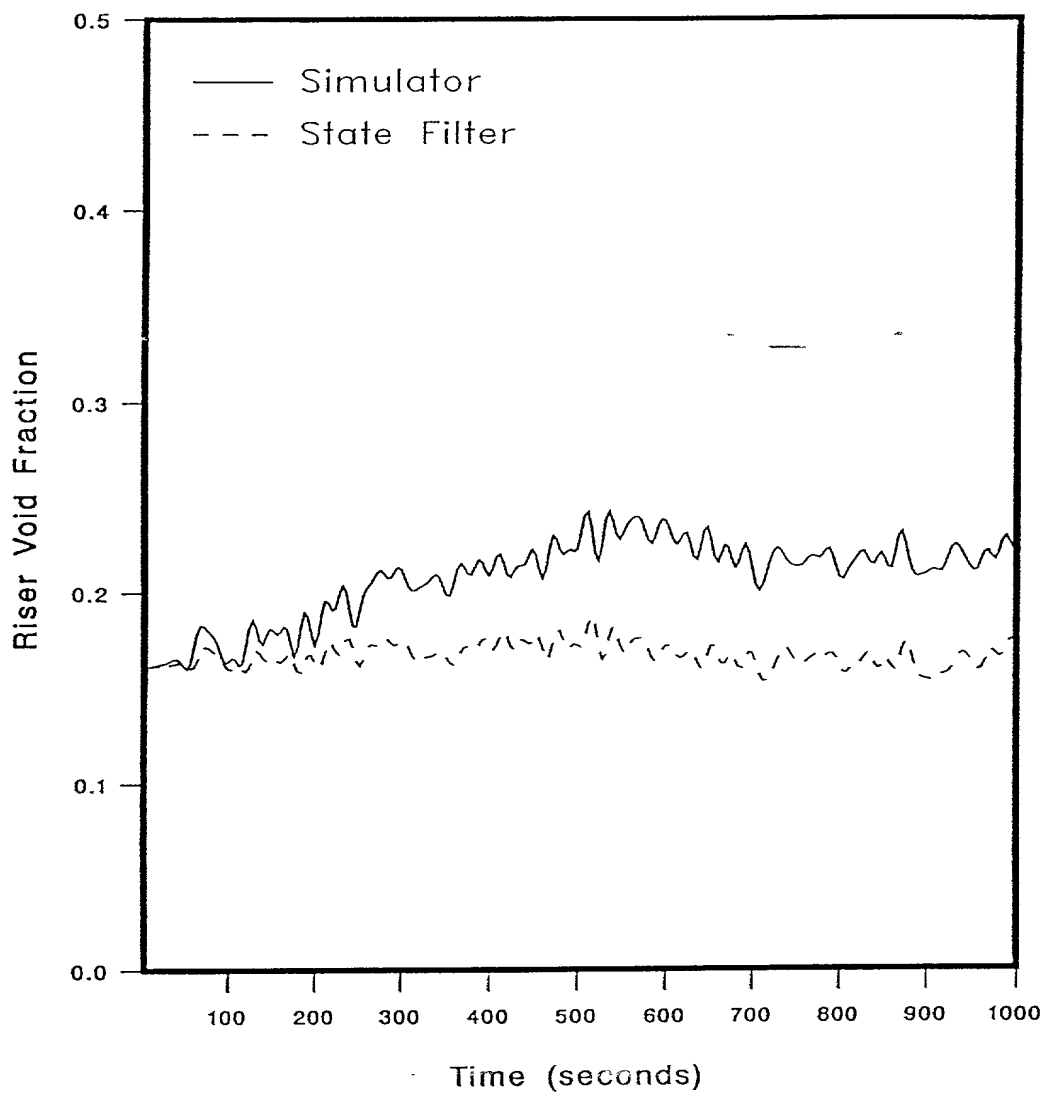


Figure 92. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (Low Noise Environment).

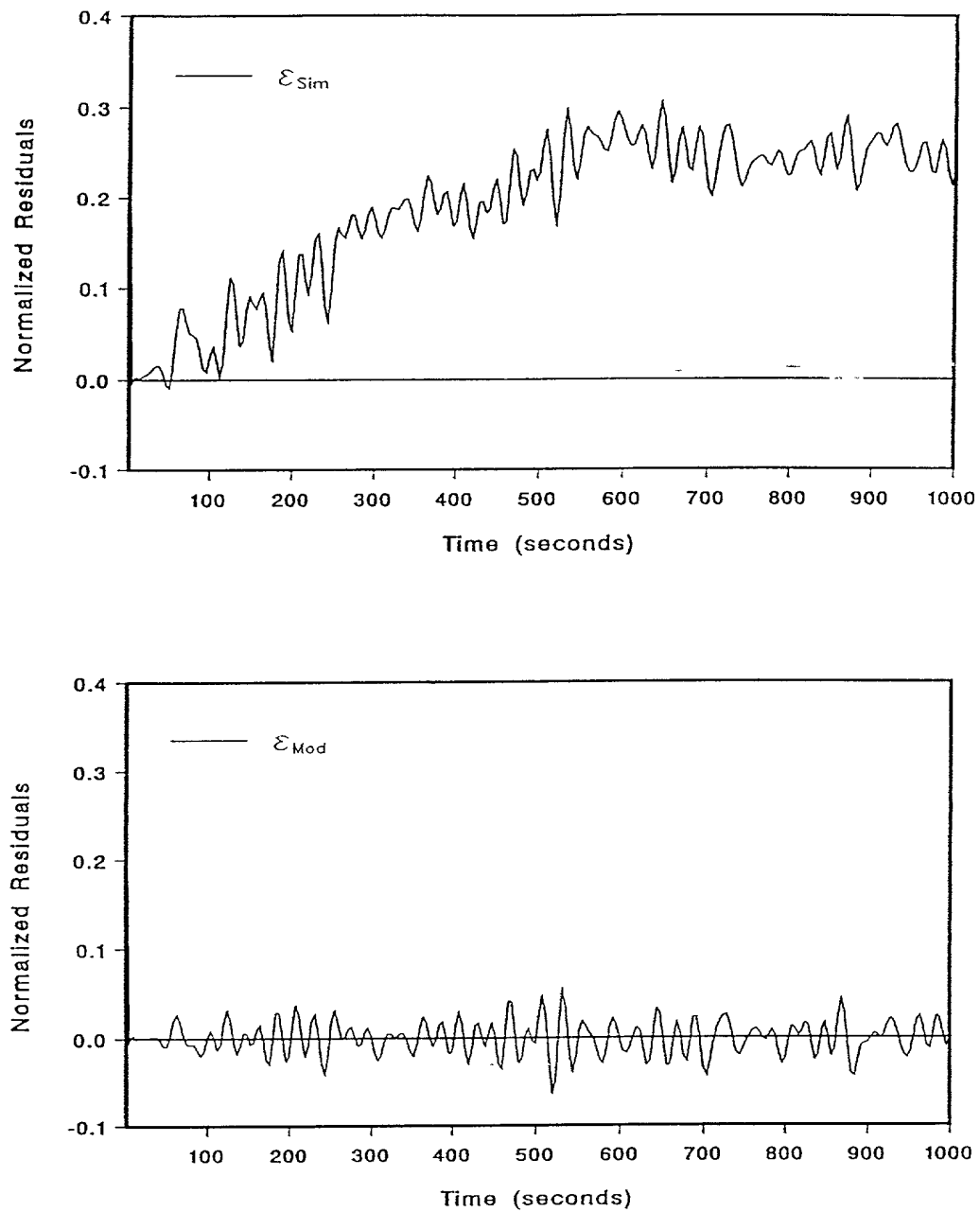


Figure 93. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 with a Ramp Input (Low Noise Environment).

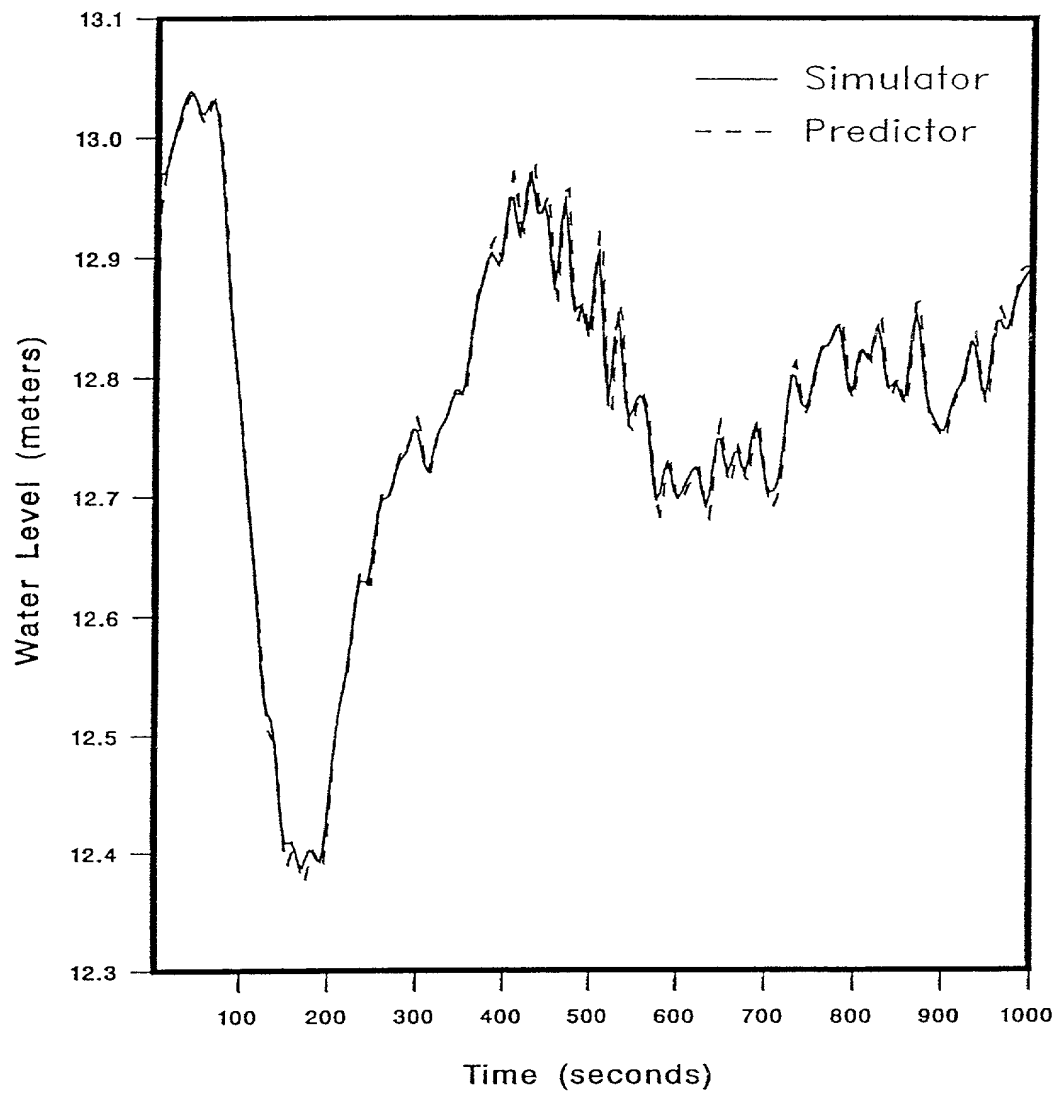


Figure 94. UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (Low Noise Environment).

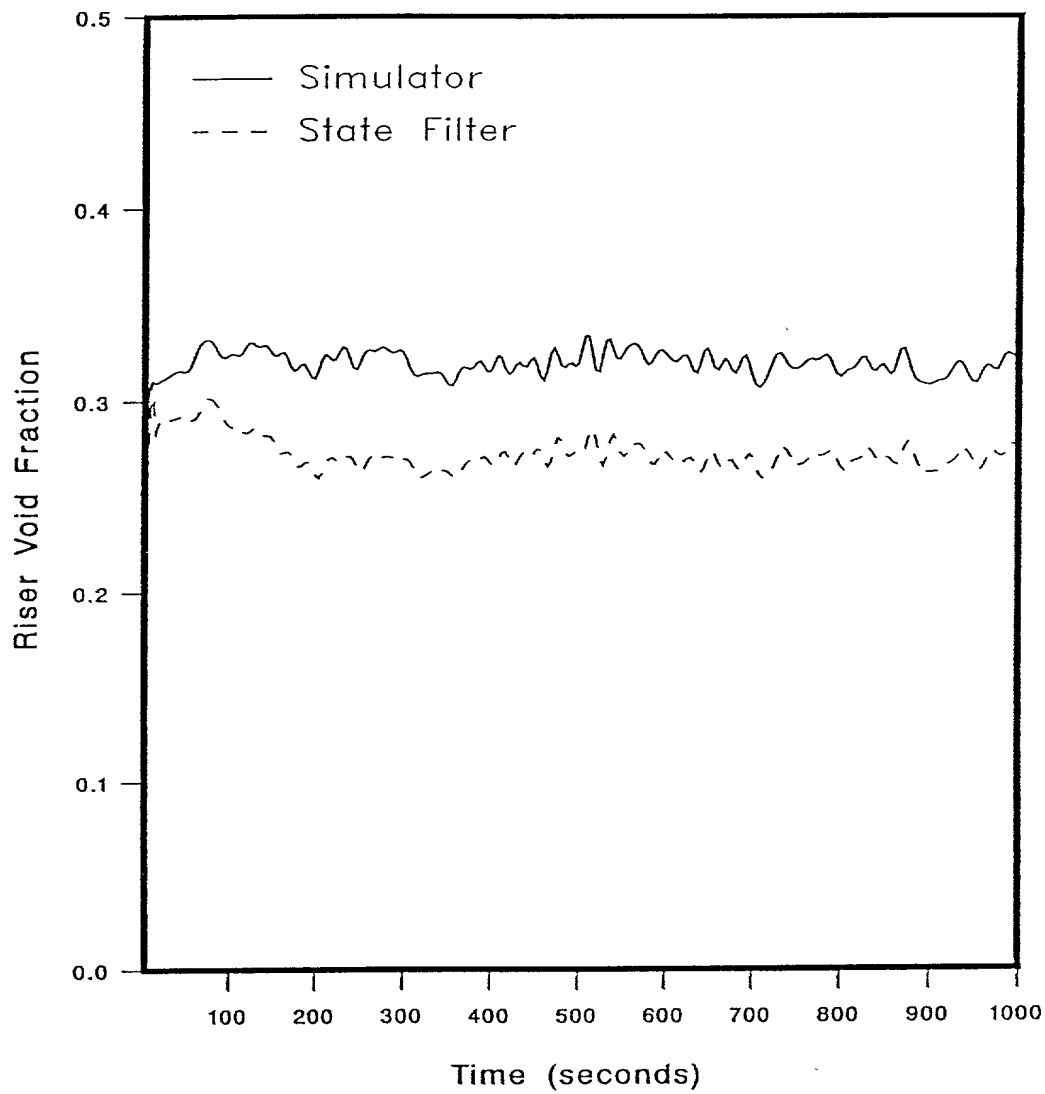


Figure 95. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (Low Noise Environment).

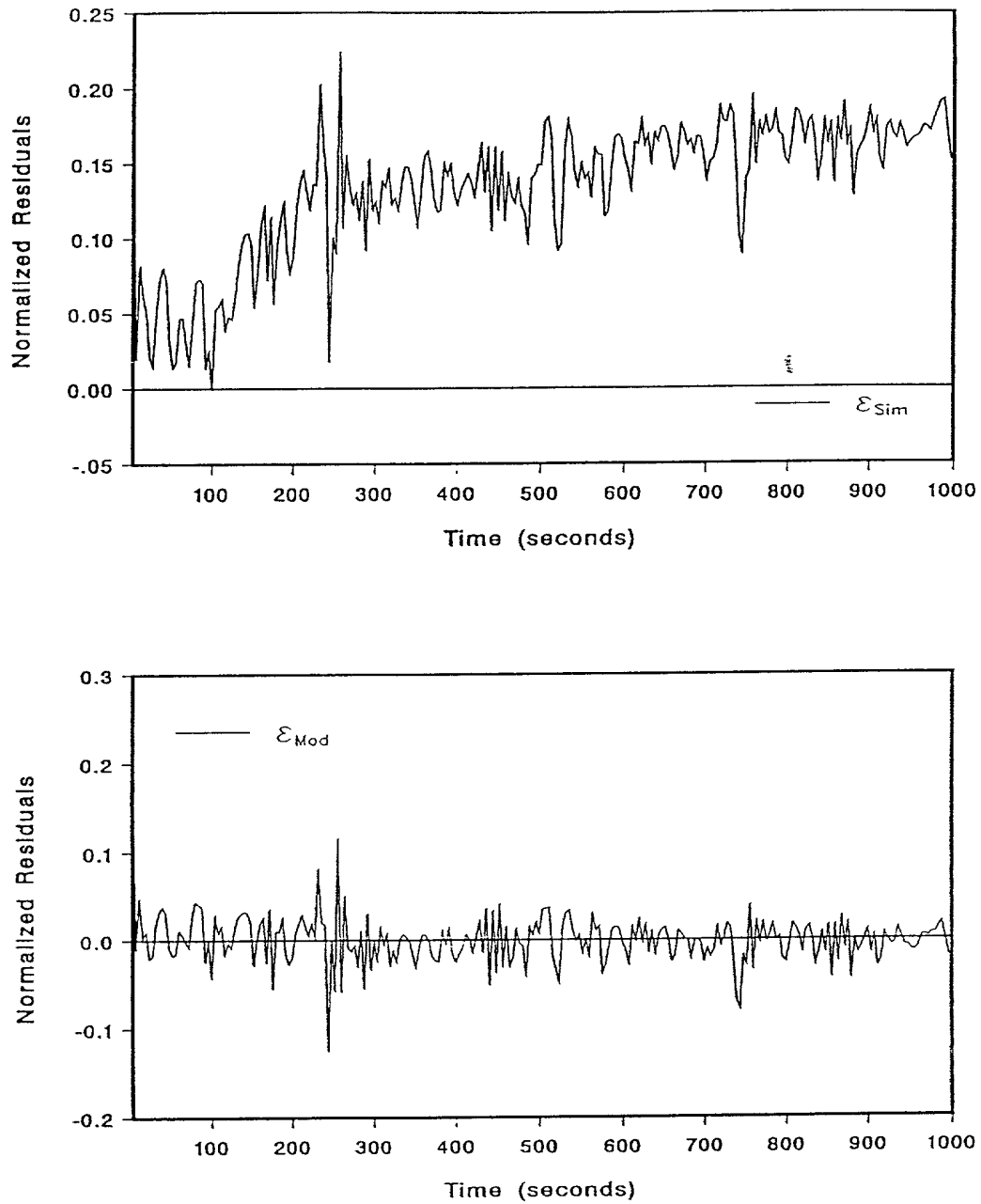


Figure 96. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (Low Noise Environment).

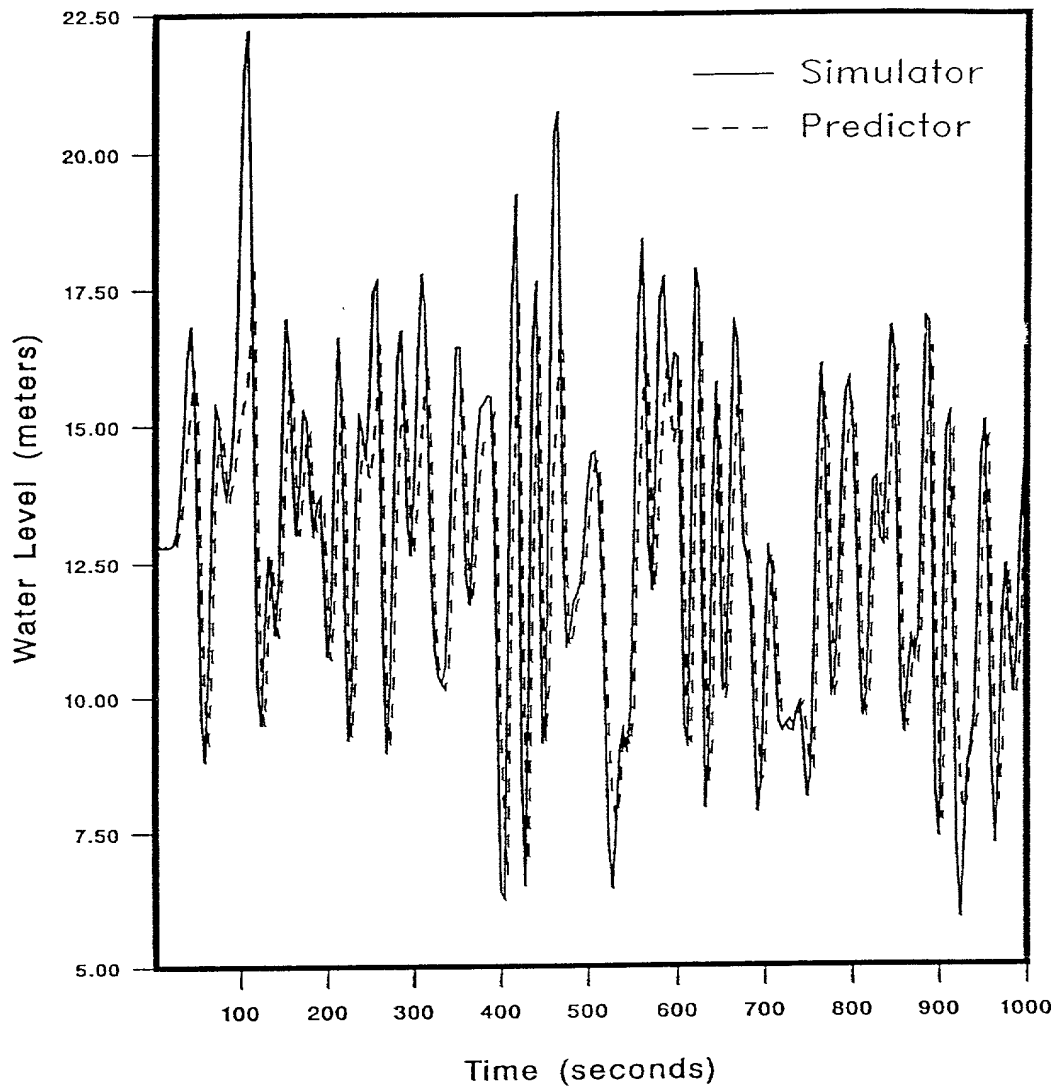


Figure 97. UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (High Noise Environment).

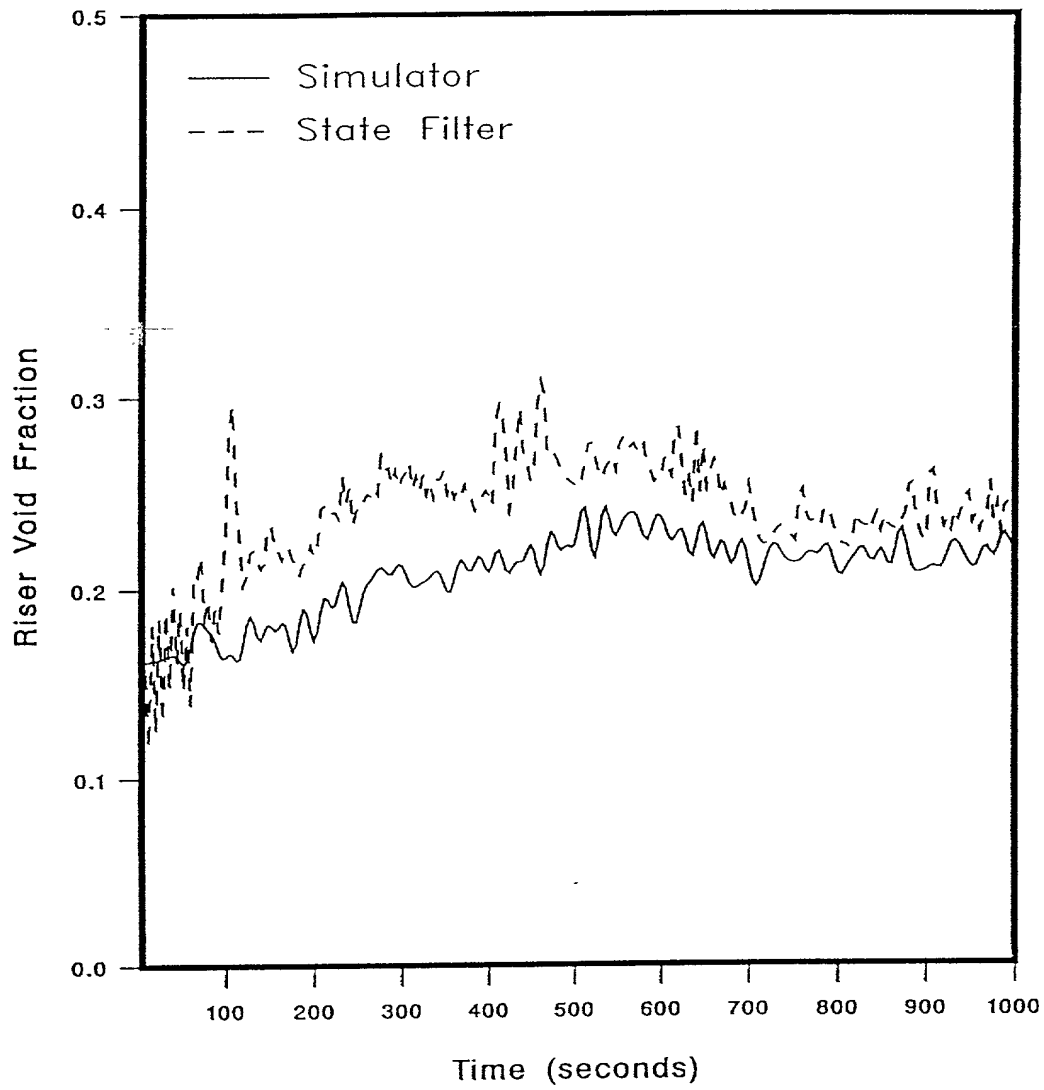


Figure 98. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (High Noise Environment).

NMSE for the state filtered values is 4.4%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 99. The average $\epsilon_{\text{Sim}}(t)$ value is -12.1% and the the average $\epsilon_{\text{Mod}}(t)$ value is -1.2%.

The UTSG process scheduled model (predictor) response to the step input is shown in Figure 100. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter are shown in Figure 101. The NMSE for the state filtered values is 3.5%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 102. The average $\epsilon_{\text{Sim}}(t)$ value is 12.1% and the the average $\epsilon_{\text{Mod}}(t)$ value is 1.3%.

The large differences between the $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$ values, in the validation data set, is due to the fact that the NN state filter was trained with the UTSG scheduled model state information, which is only moderately accurate.

VII.5.2 Filter 2 - Hybrid Adaptive Filter for the Riser Void Fraction

The performance of Filter 2 on the validation data set is evaluated. The UTSG process simulator model (predictor) response to the ramp input is shown in Figure 103. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter to the ramp input are shown in Figure 104. The NMSE for the state filtered values is 0.25%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 105. The average $\epsilon_{\text{Sim}}(t)$ value is 4% and the the average $\epsilon_{\text{Mod}}(t)$ value is 1%.

The UTSG process simulator model (predictor) response to the step input is shown in Figure 106. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter are shown in Figure 107. The NMSE for the state filtered values is 0.05%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 108. The average $\epsilon_{\text{Sim}}(t)$ value is 2.1% and the the average $\epsilon_{\text{Mod}}(t)$ value is 0.8%.

Now, a higher level of process and measurement noise, zero-mean, white, Gaussian with 0.15 sd, is added to the validation data set and the performance of Filter 2 is evaluated. The UTSG process simulator model (predictor) response to the ramp input is shown in Figure 109. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter to the ramp input are shown in Figure 110. The

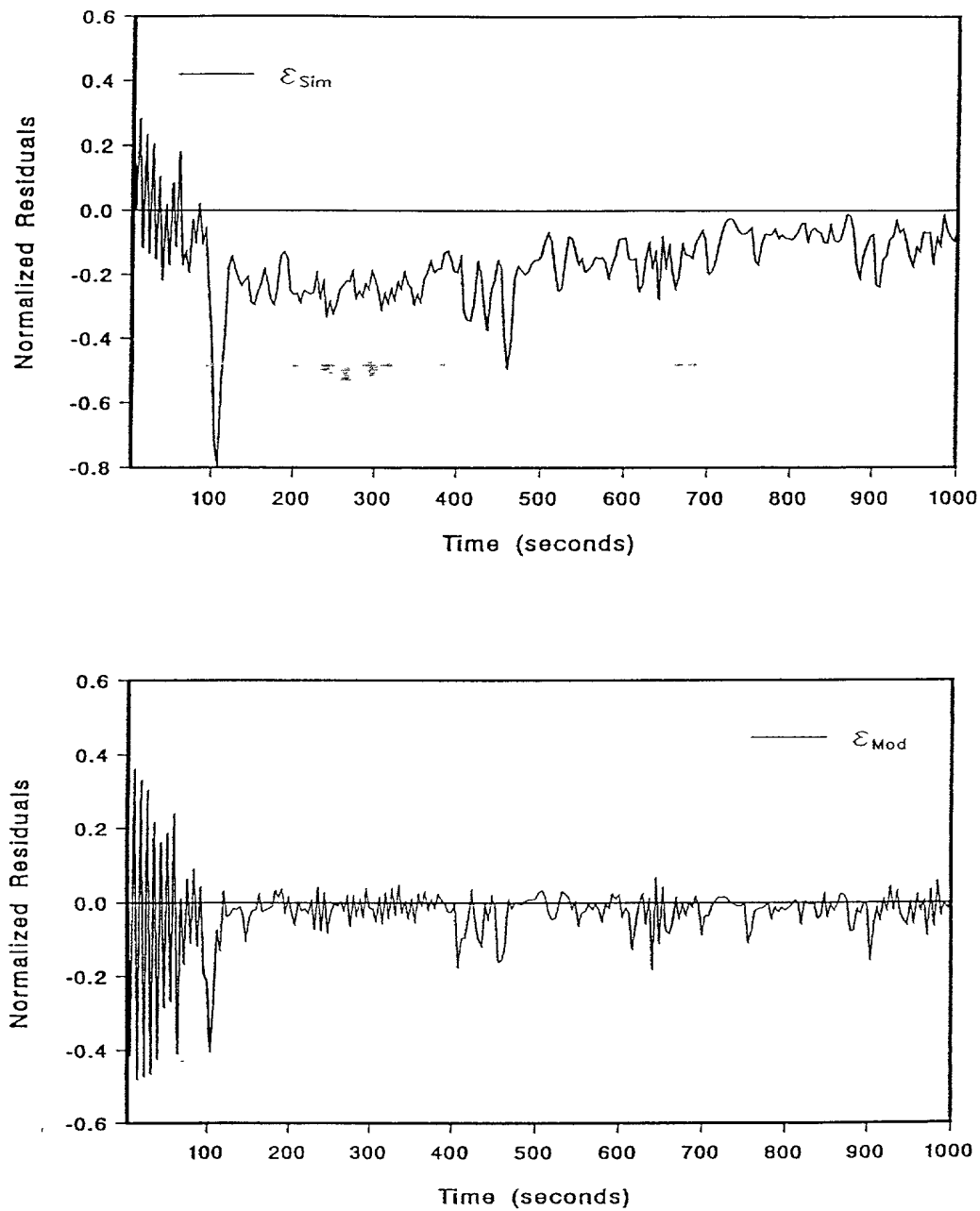


Figure 99. UTSG Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Ramp Input (High Noise Environment).

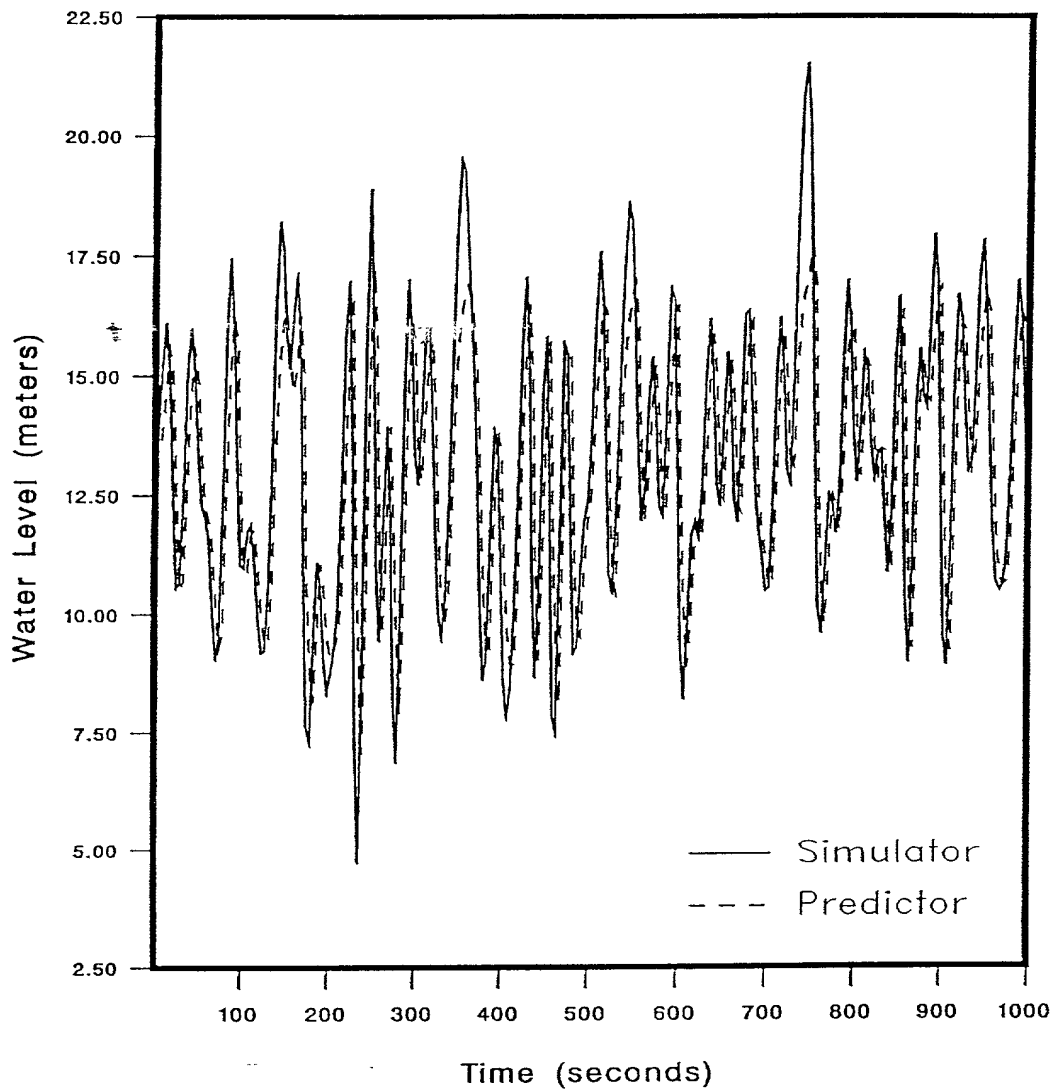


Figure 100. UTSG Process Water Level Response, from the Simulator and Scheduled Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (High Noise Environment).

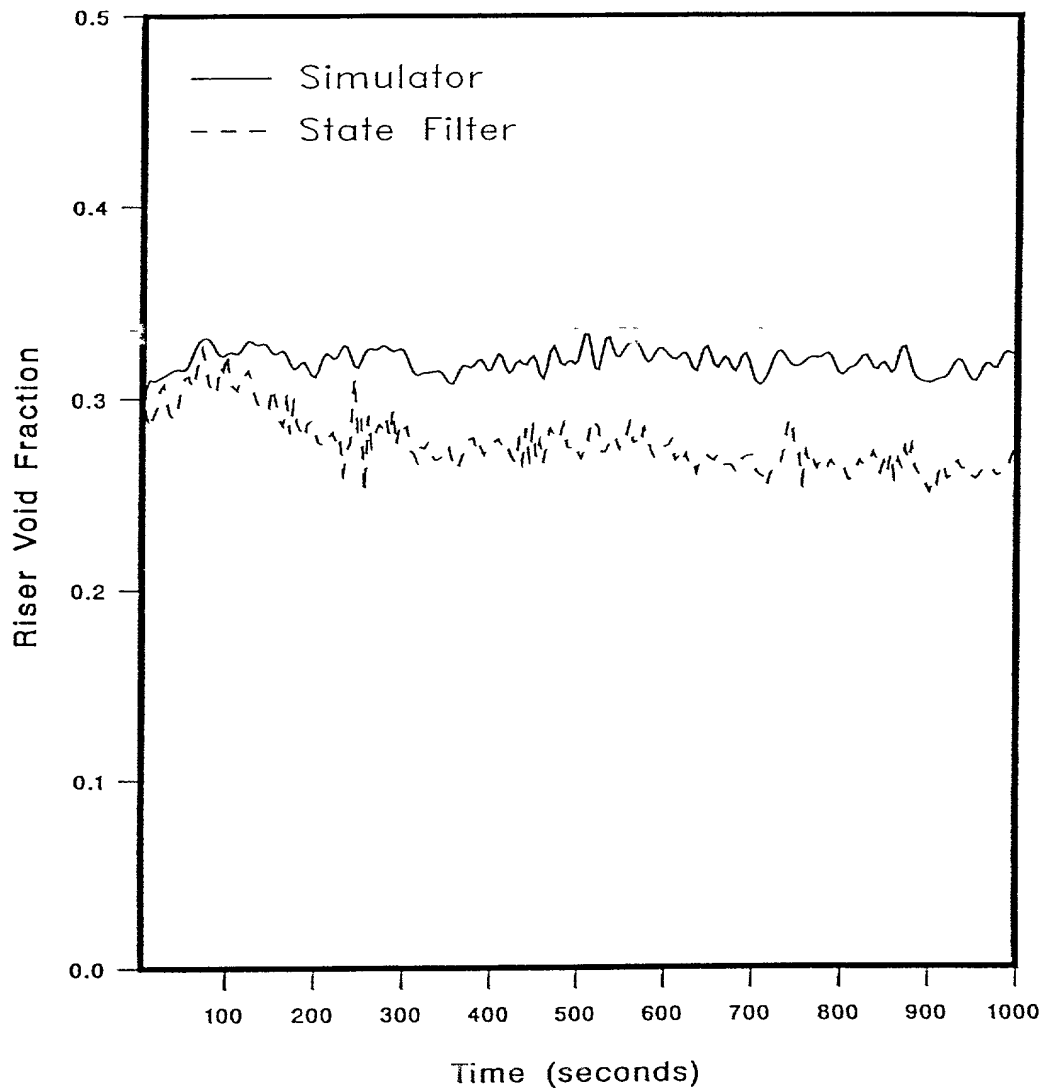


Figure 101. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (High Noise Environment).

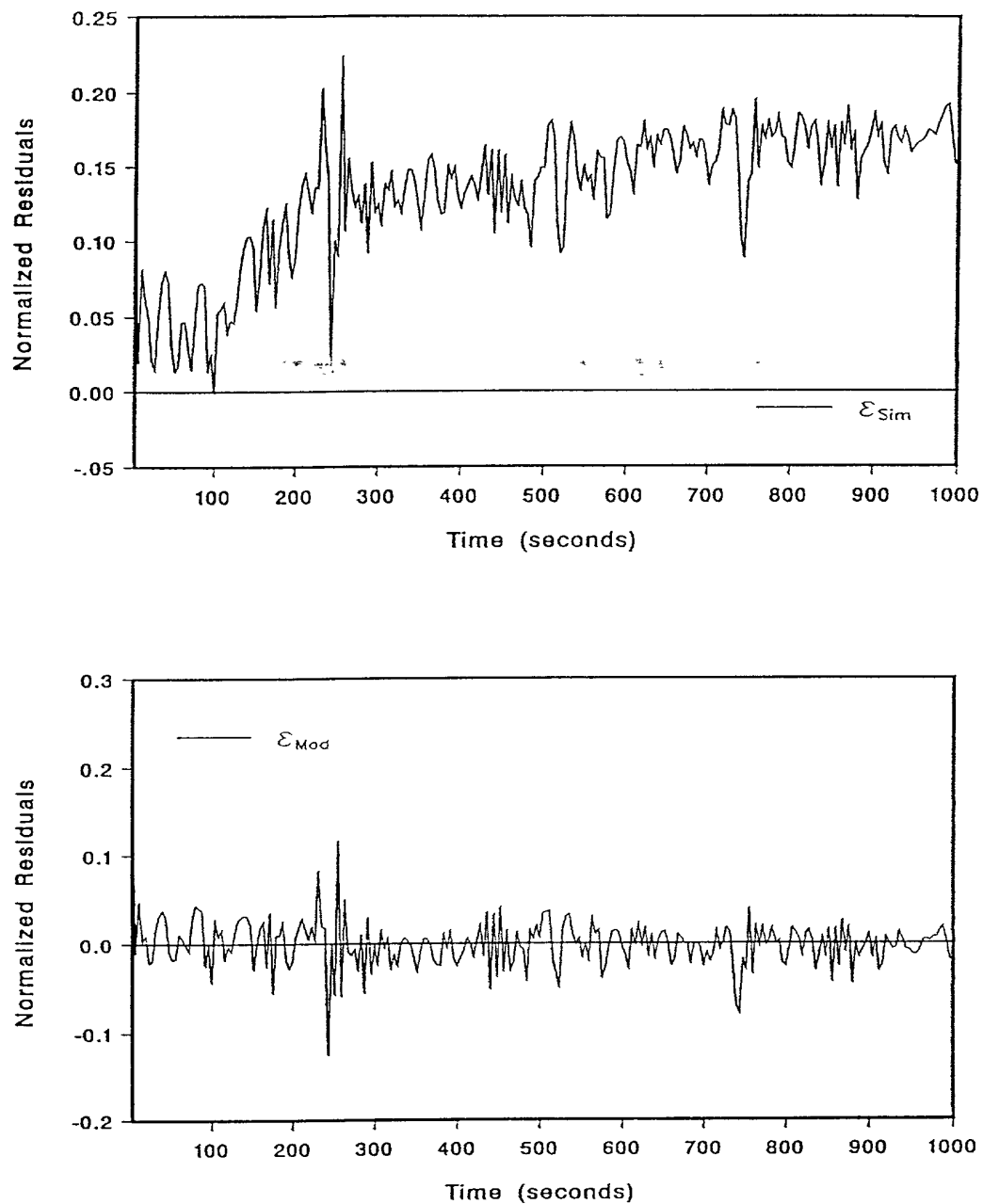


Figure 102. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 1 Using a Step Input (High Noise Environment).

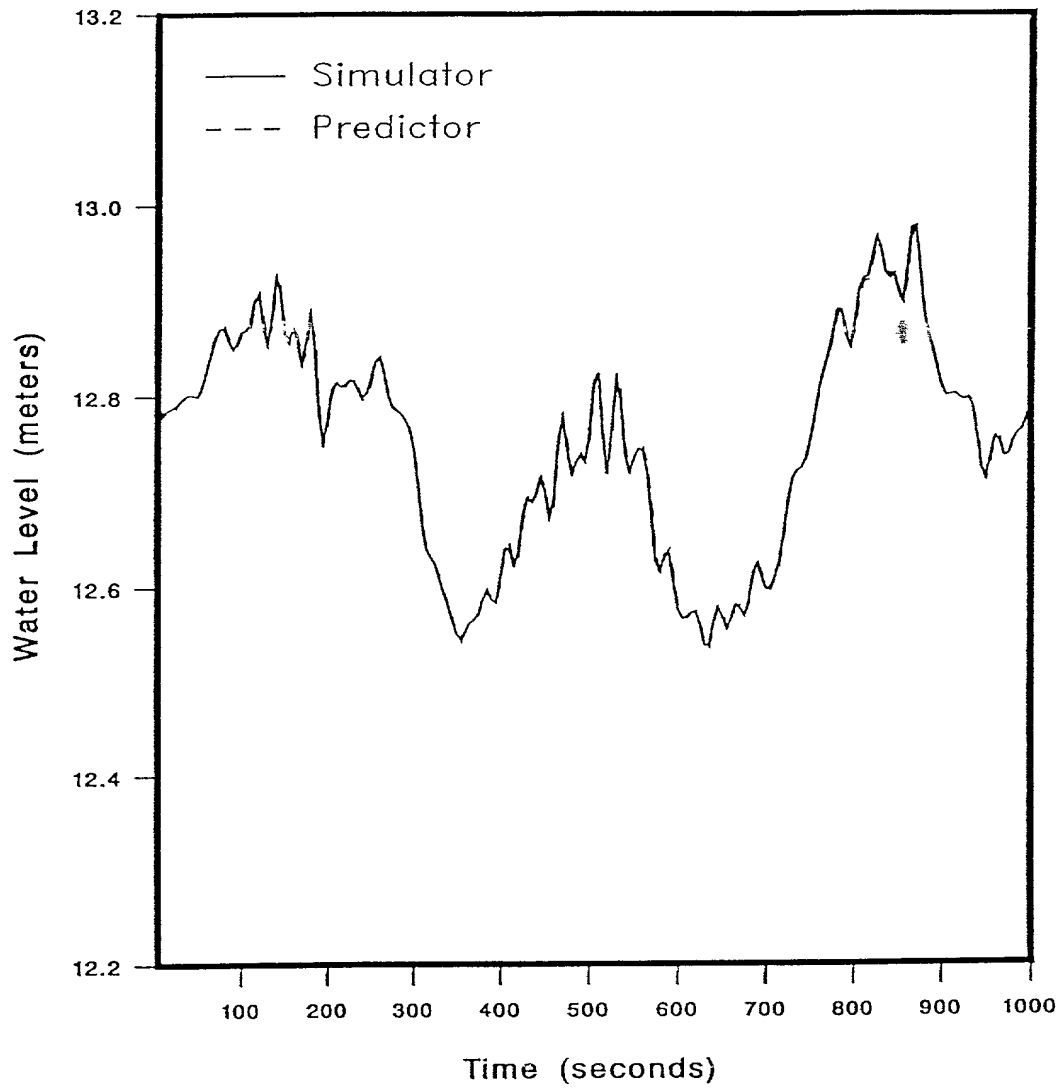


Figure 103. UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (Low Noise Environment).

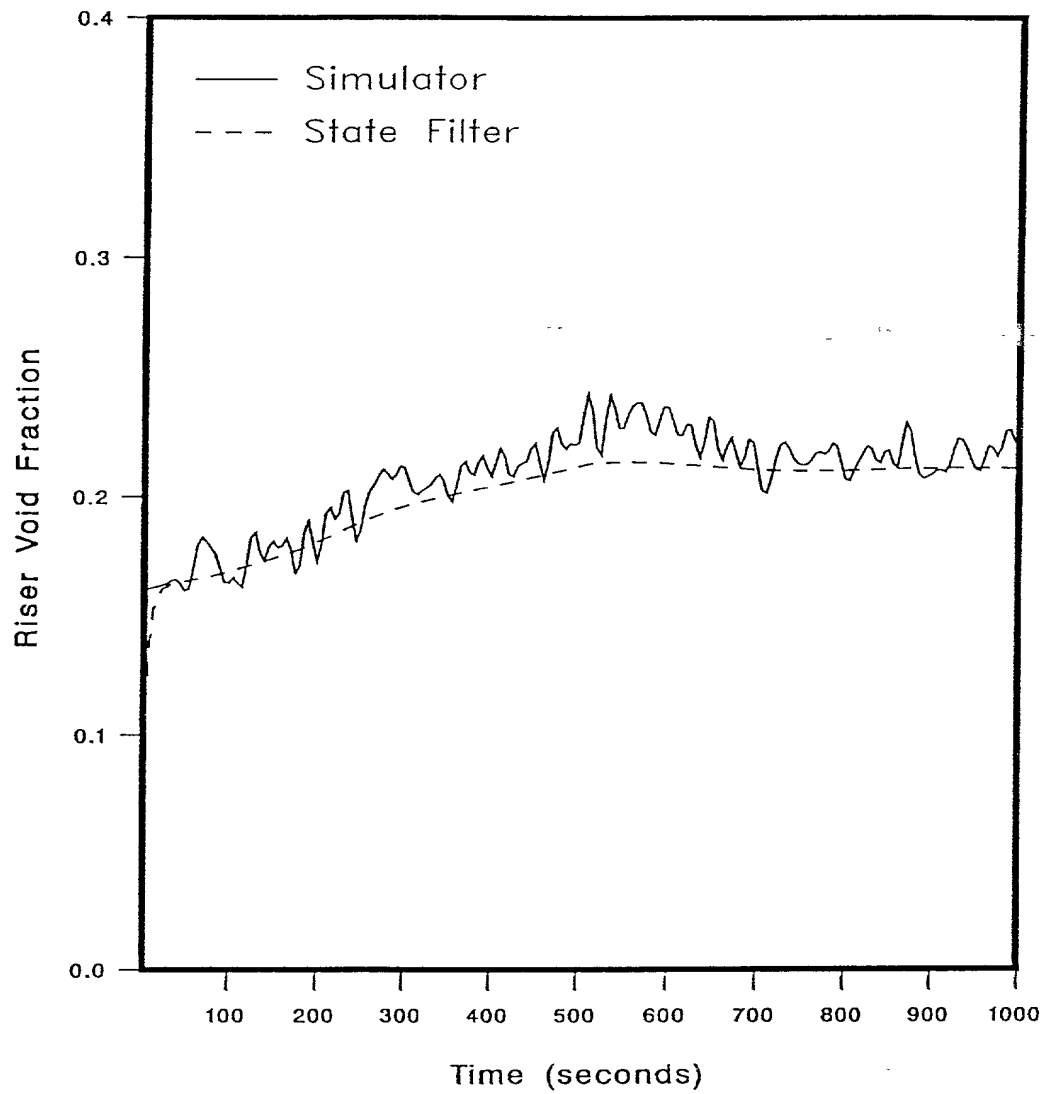


Figure 104. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (Low Noise Environment).

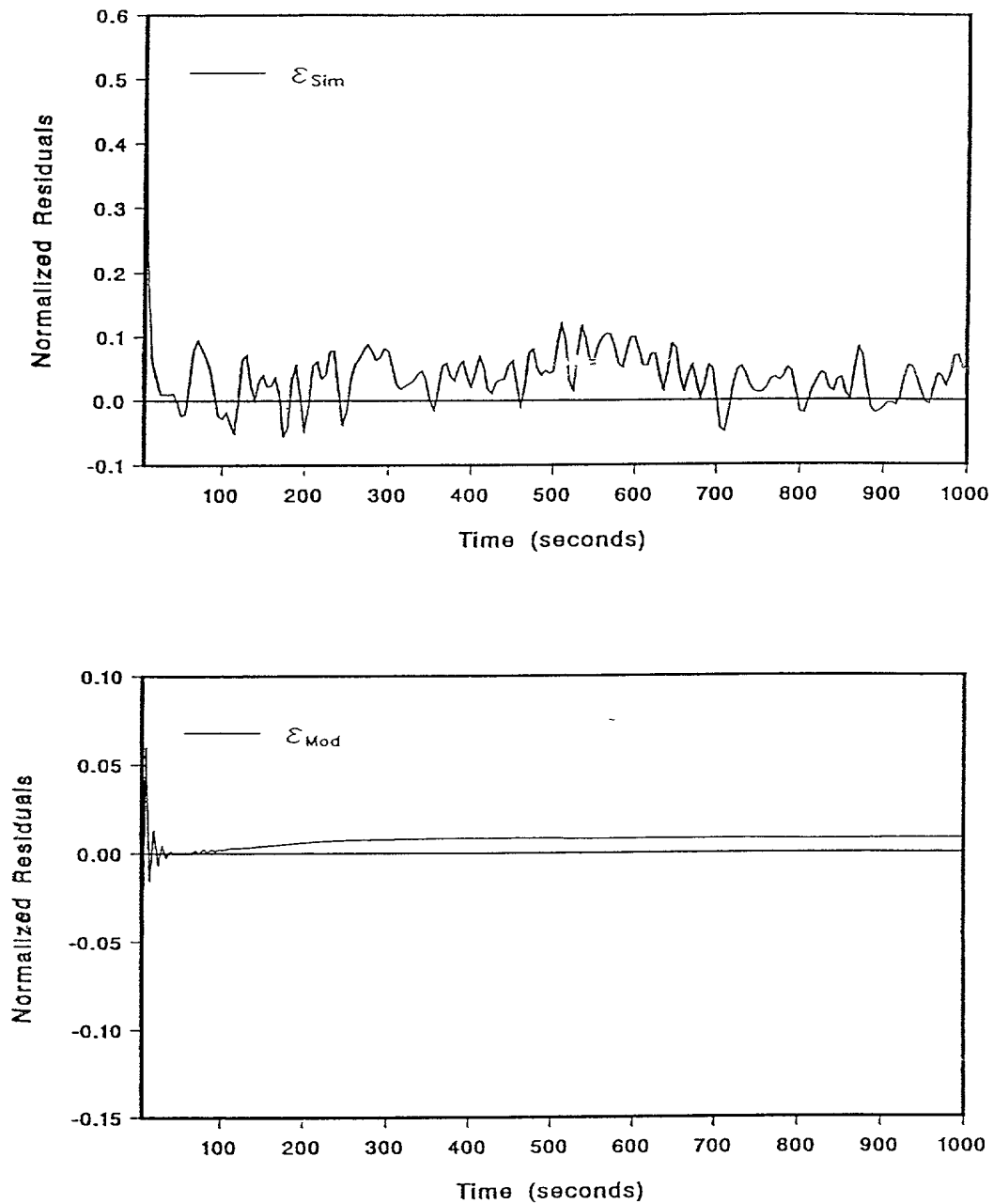


Figure 105. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (Low Noise Environment).

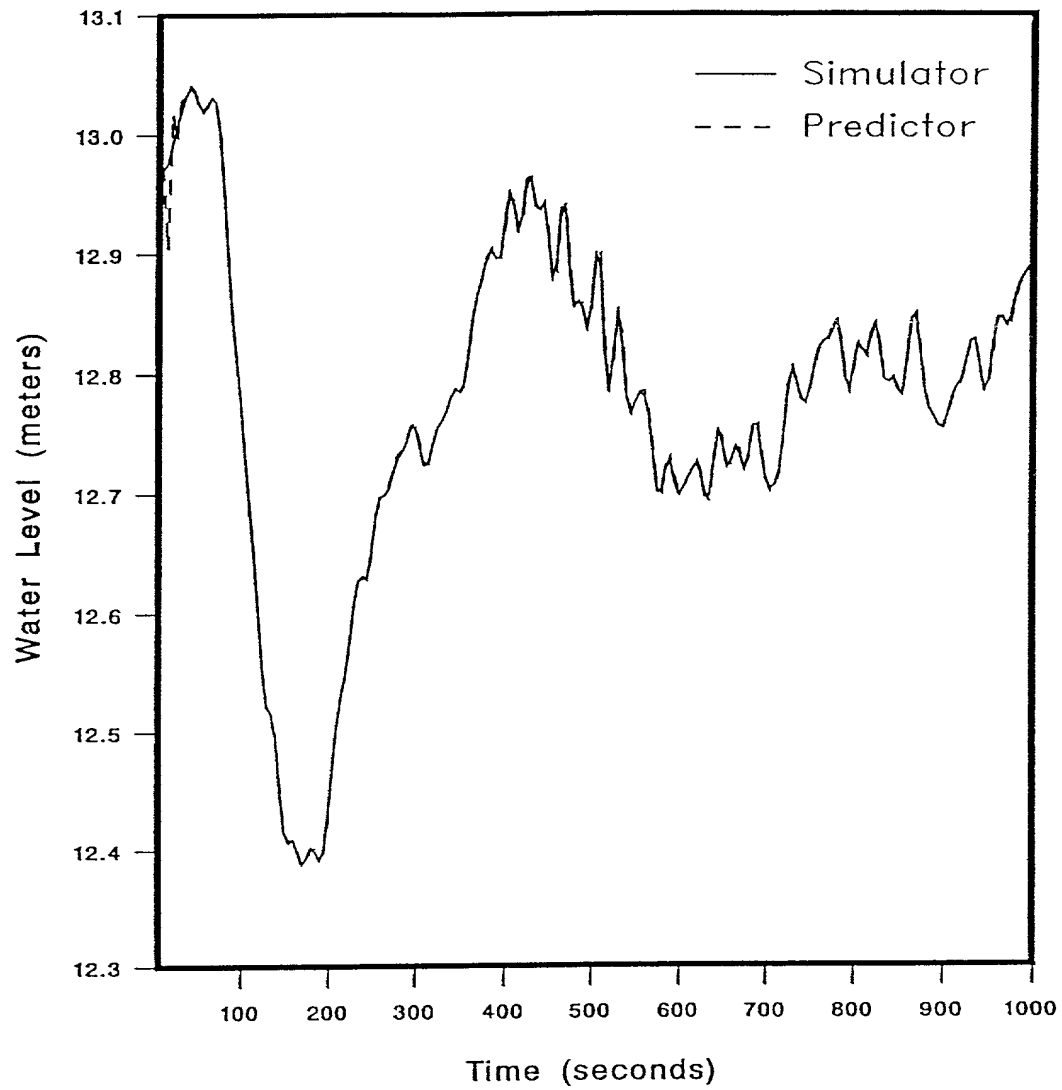


Figure 106. UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (Low Noise Environment).

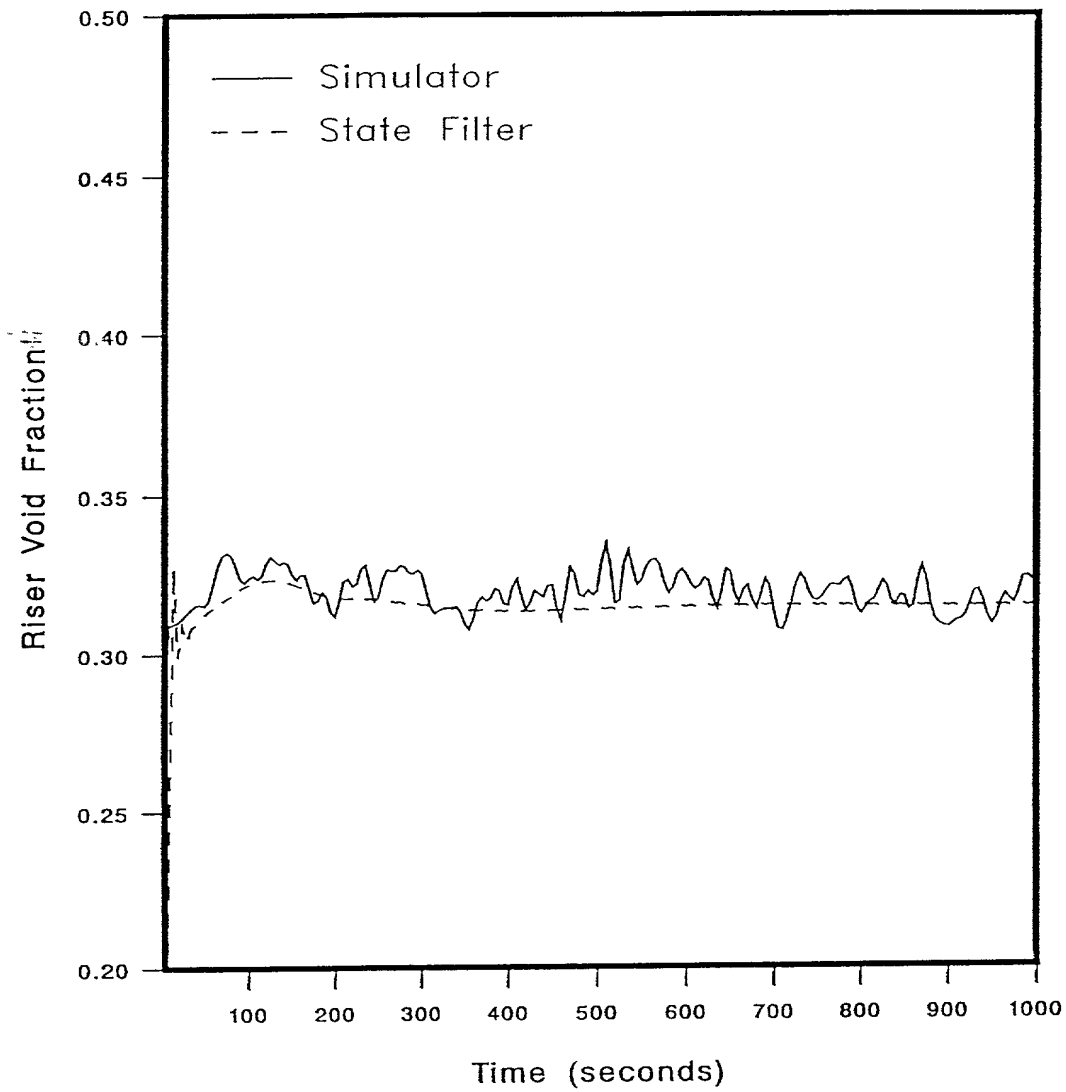


Figure 107. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (Low Noise Environment).

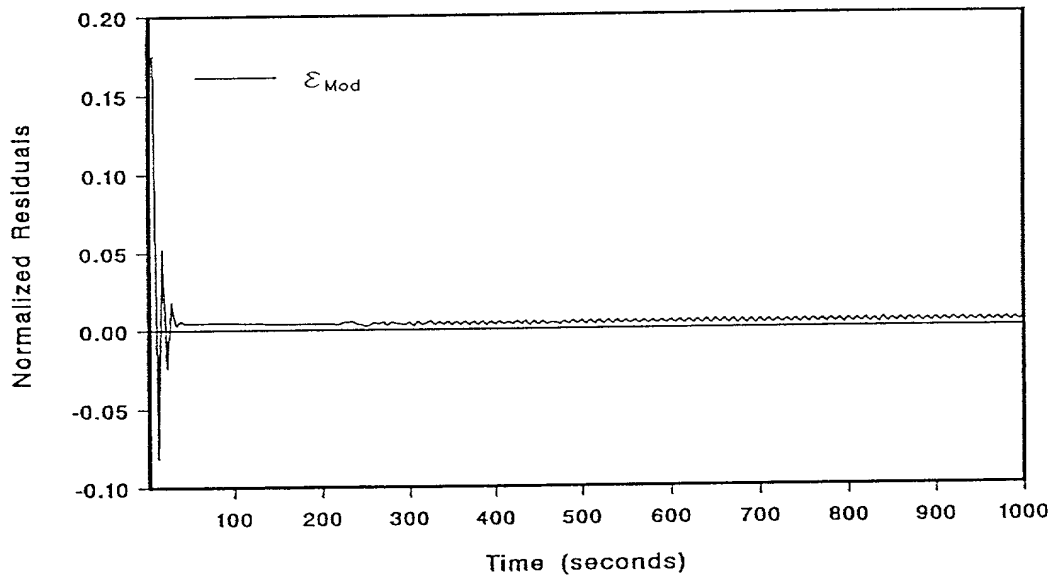
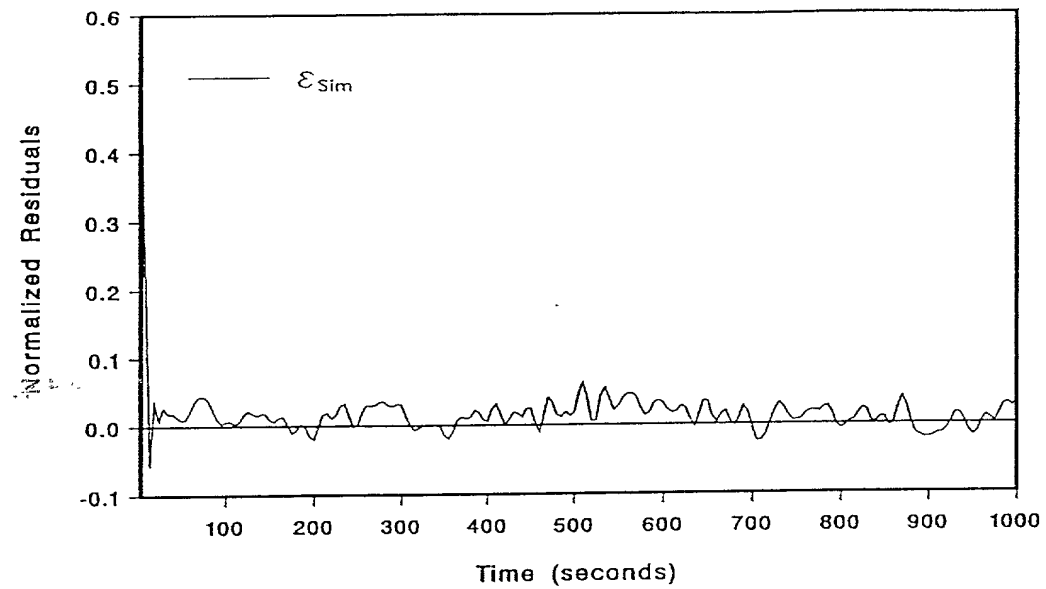


Figure 108. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (Low Noise Environment).

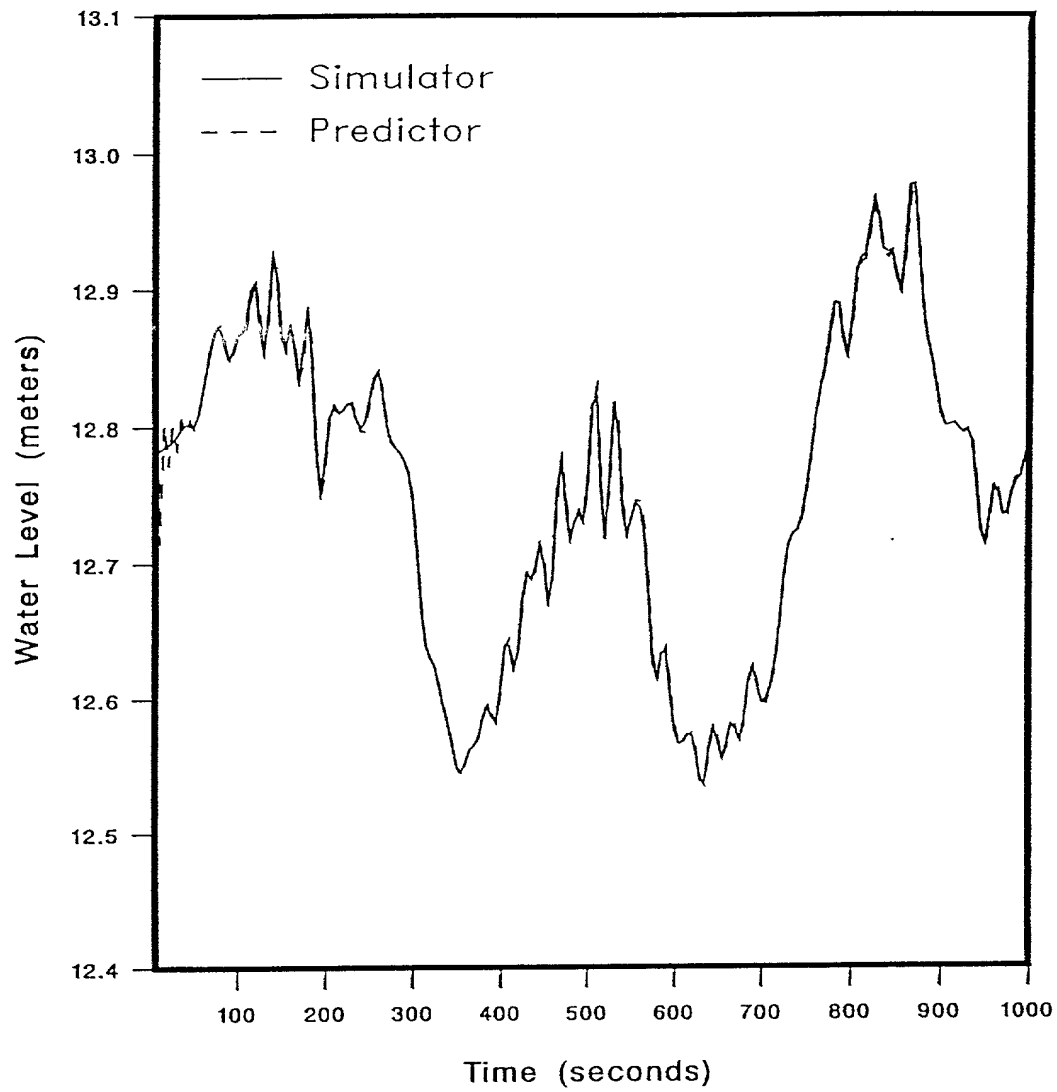


Figure 109. UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (High Noise Environment).

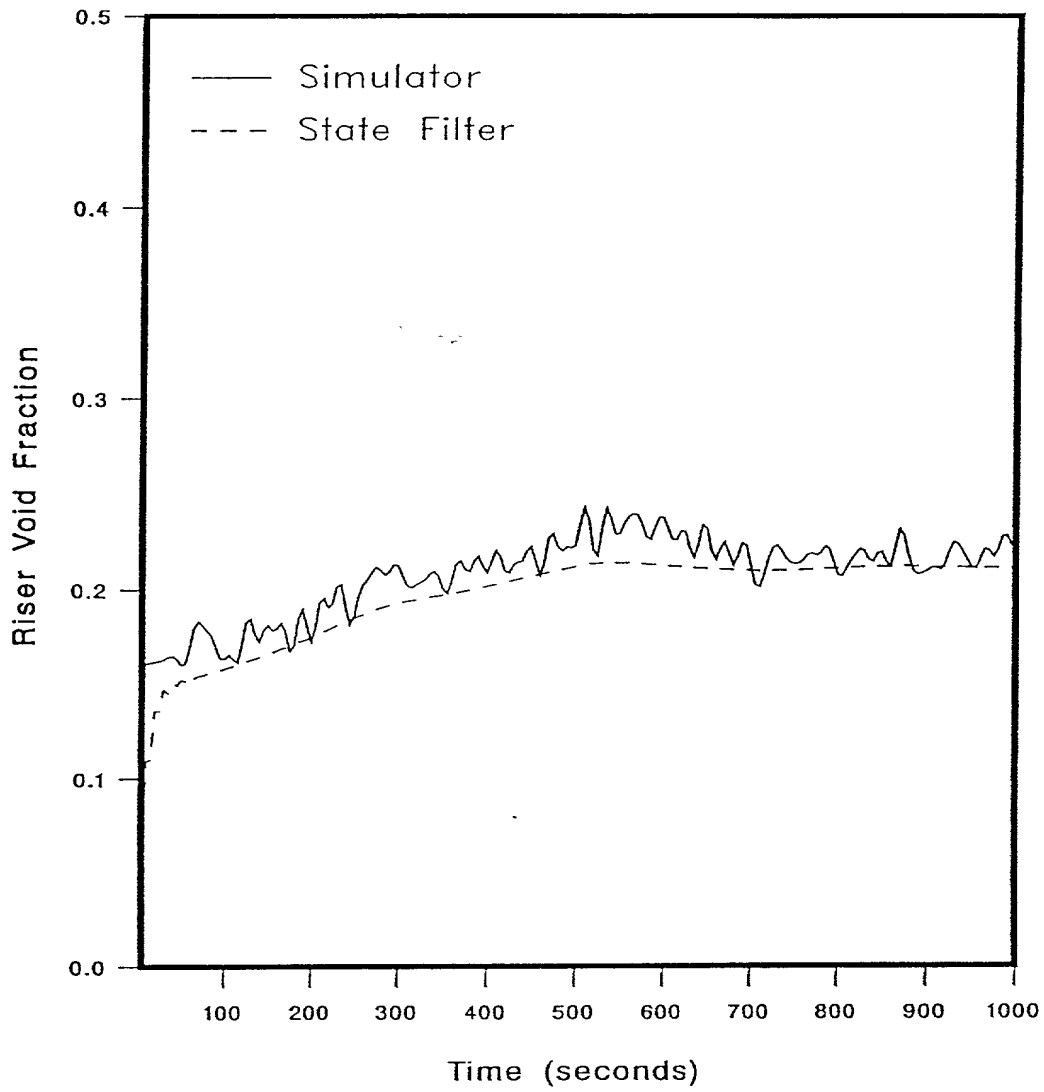


Figure 110. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (High Noise Environment).

NMSE for the state filtered values is 1.1%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 111. The average $\epsilon_{\text{Sim}}(t)$ value is 8.3% and the the average $\epsilon_{\text{Mod}}(t)$ value is 1.6%.

The UTSG process simulator model (predictor) response to the step input is shown in Figure 112. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter are shown in Figure 113. The NMSE for the state filtered values is 0.2%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 114. The average $\epsilon_{\text{Sim}}(t)$ value is 3.6% and the the average $\epsilon_{\text{Mod}}(t)$ value is 0.9%.

The differences between the $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, for the ramp and step inputs in the validation data set, is due to the fact that the NN state filter was trained with the simulator model state information, which is close to the actual process but not exact. However, the UTSG simulator model is far more accurate than the scheduled model (used in Filter 1); therefore the NN state filter in Filter 2 is much more accurate than the NN state filter in Filter 1.

VII.5.3 Filter 3 - Adaptive Filter for the Riser Void Fraction

The performance of Filter 3 on the validation data set is evaluated. The NN output predictor response to the ramp input is shown in Figure 115. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter to the ramp input are shown in Figure 116. The NMSE for the state filtered values is 0.56%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 117. The average $\epsilon_{\text{Sim}}(t)$ value is 0.38% and the the average $\epsilon_{\text{Mod}}(t)$ value is 0.34%.

The NN output predictor response to the step input is shown in Figure 118. The riser void fraction filtered values, $\hat{\alpha}_{\text{NN},R}(t|t)$, as determined by the NN state filter are shown in Figure 119. The NMSE for the state filtered values is 0.05%. The normalized residuals, $\epsilon_{\text{Sim}}(t)$ and $\epsilon_{\text{Mod}}(t)$, are shown in Figure 120. The average $\epsilon_{\text{Sim}}(t)$ value is 0.18% and the the average $\epsilon_{\text{Mod}}(t)$ value is 0.12%.

Now, a higher level of process and measurement noise, zero-mean, white, Gaussian with 0.15 sd, is added to the validation data set and the performance of Filter 3

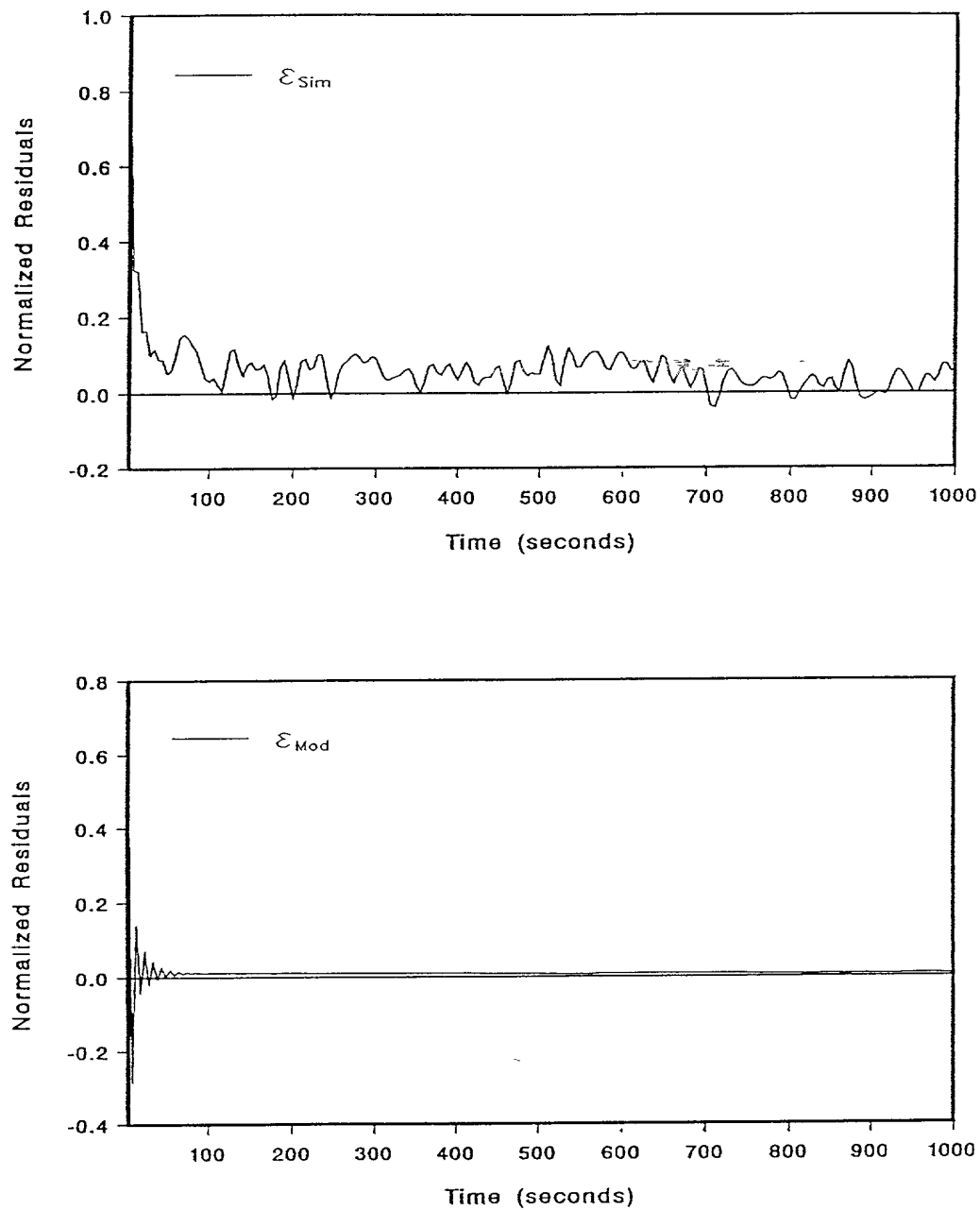


Figure 111. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Ramp Input (High Noise Environment).

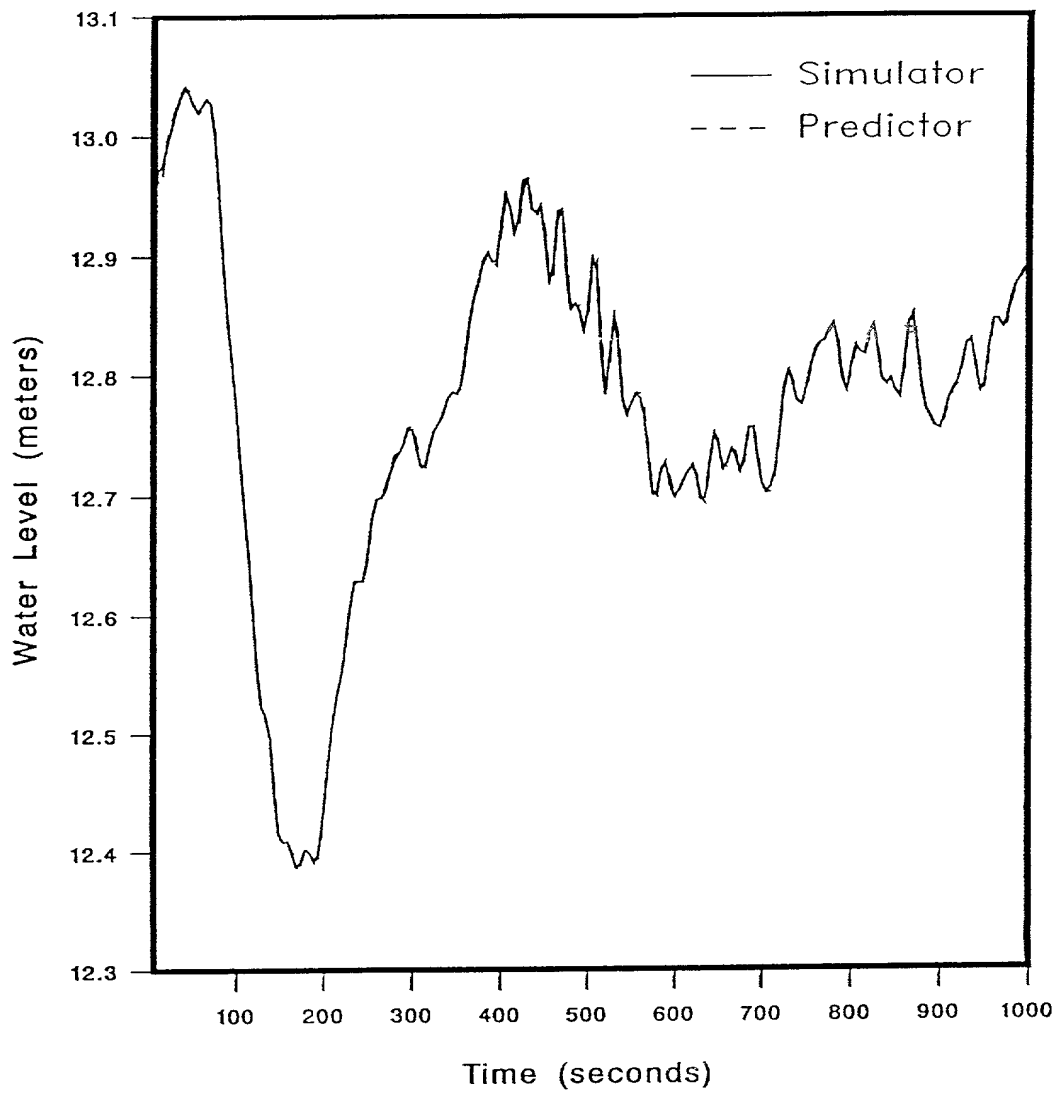


Figure 112. UTSG Process Water Level Response, from the Simulator and Simulator Model, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (High Noise Environment).

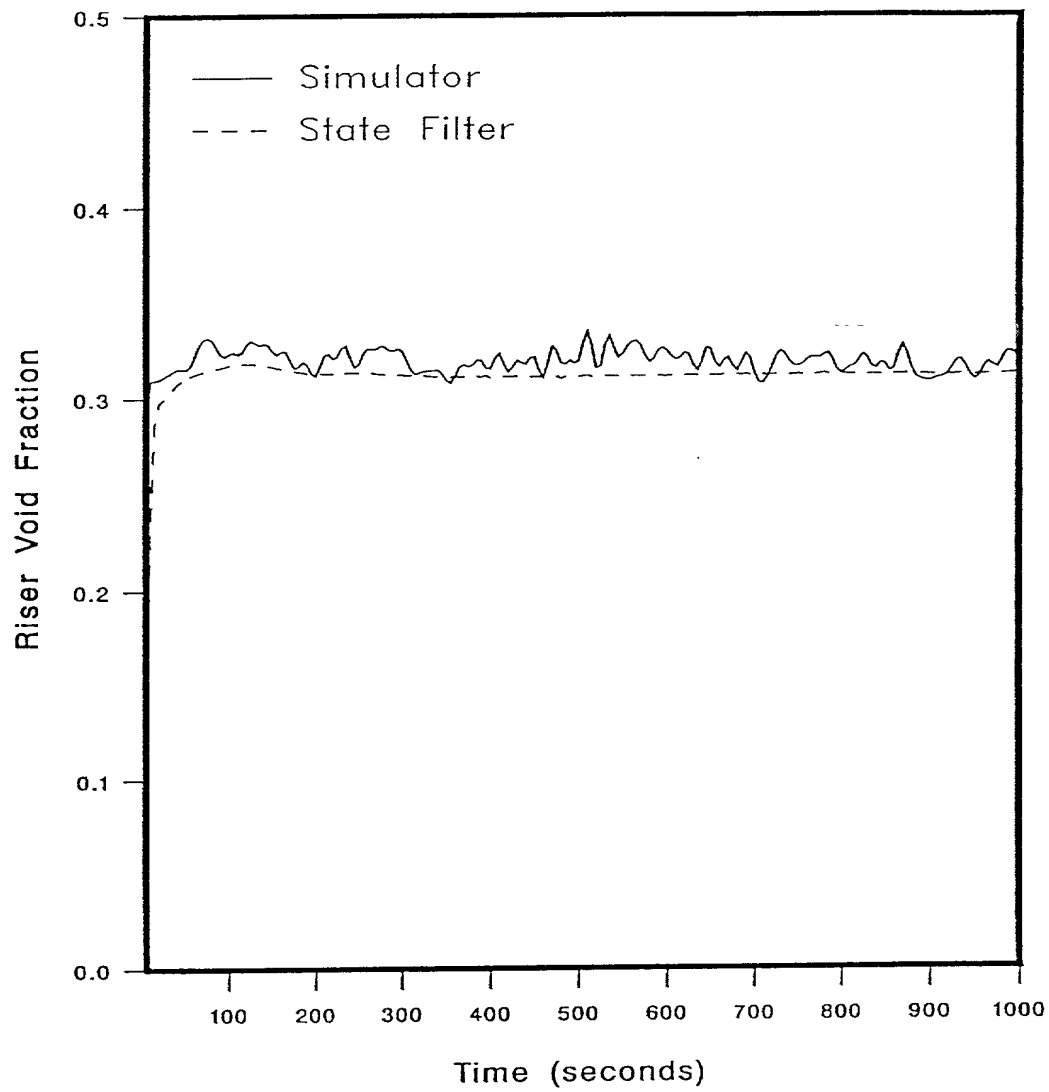


Figure 113. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (High Noise Environment).

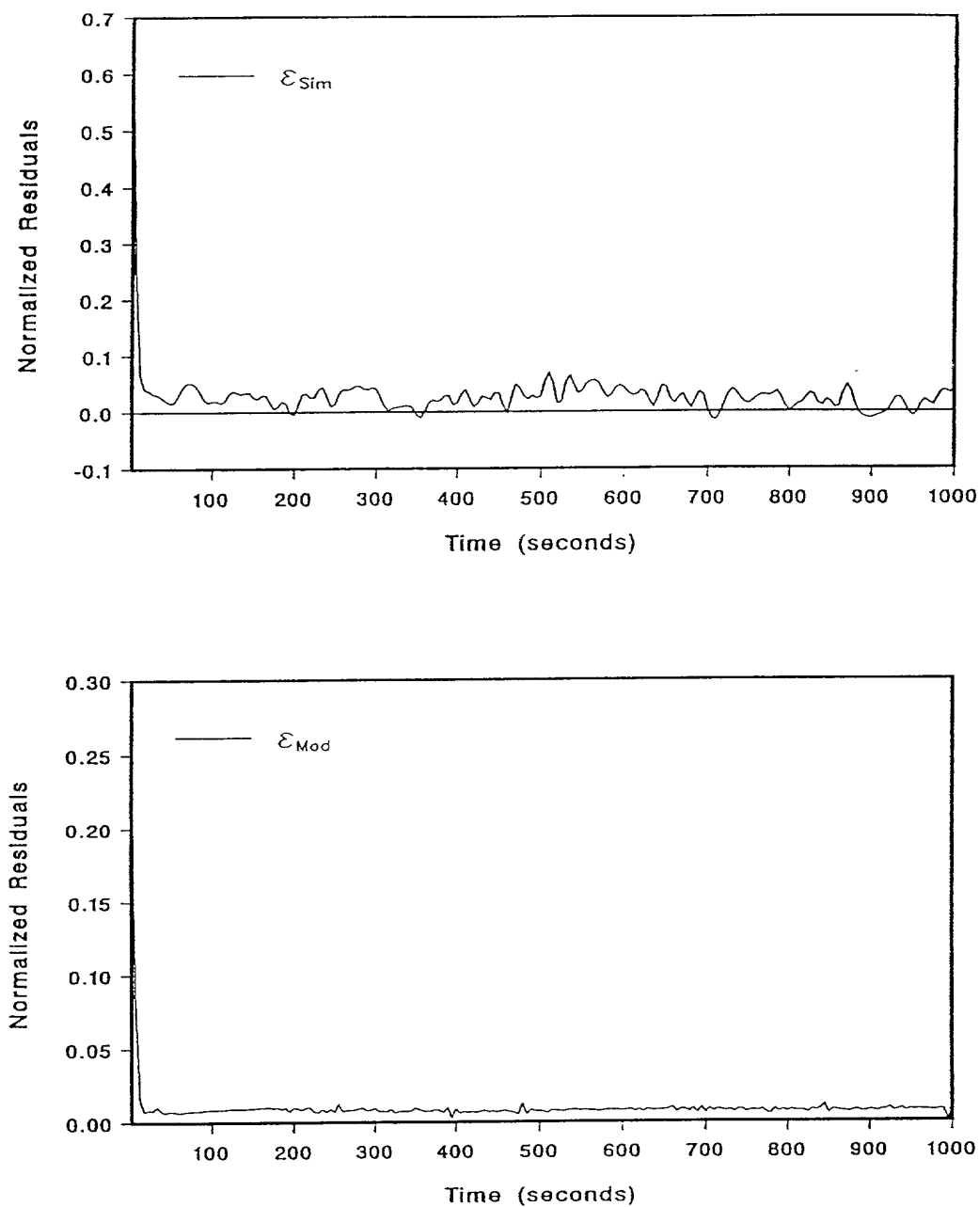


Figure 114. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Hybrid Adaptive NN Riser Void Fraction Filter 2 Using a Step Input (High Noise Environment).

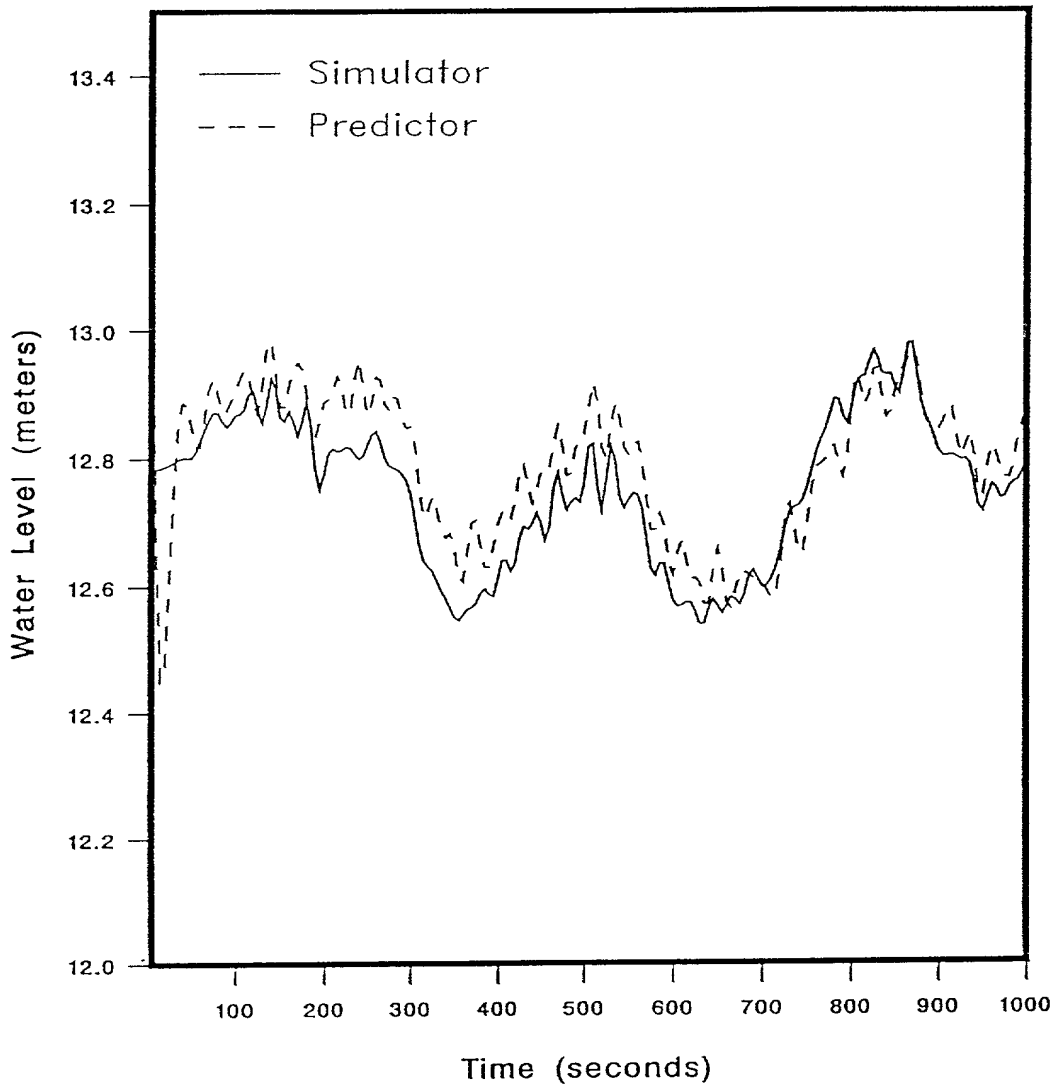


Figure 115. UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (Low Noise Environment).

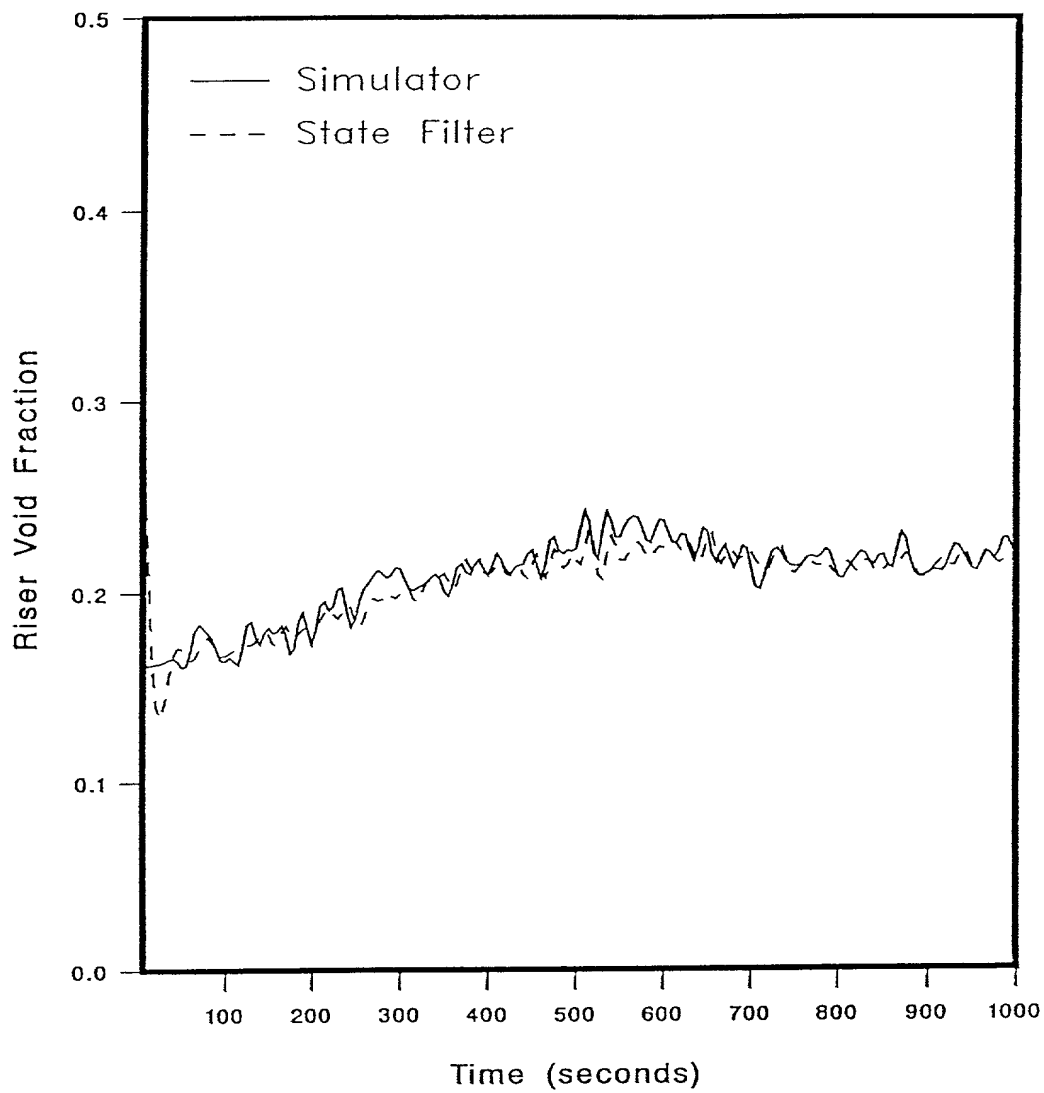


Figure 116. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (Low Noise Environment).

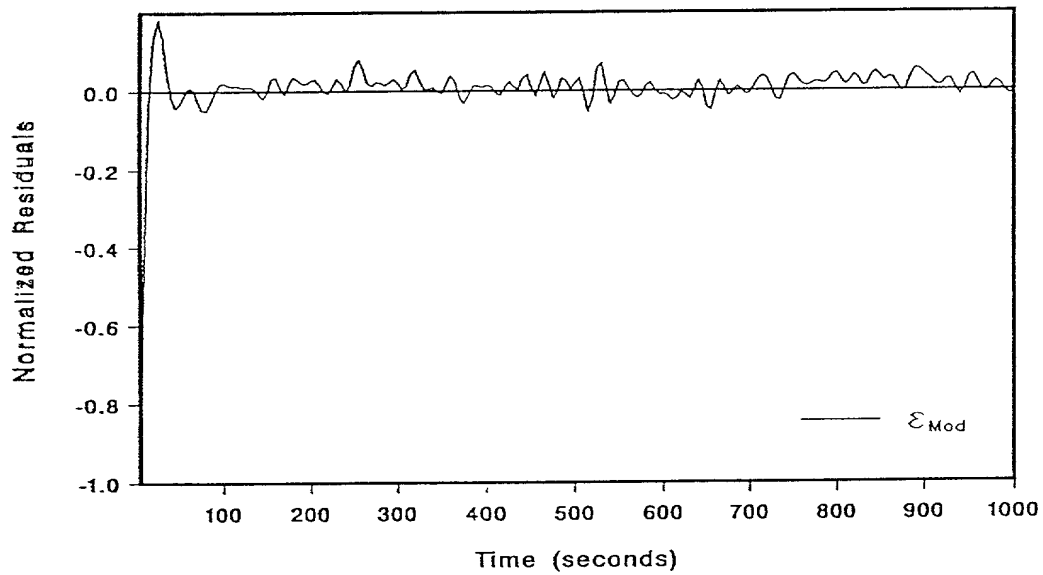
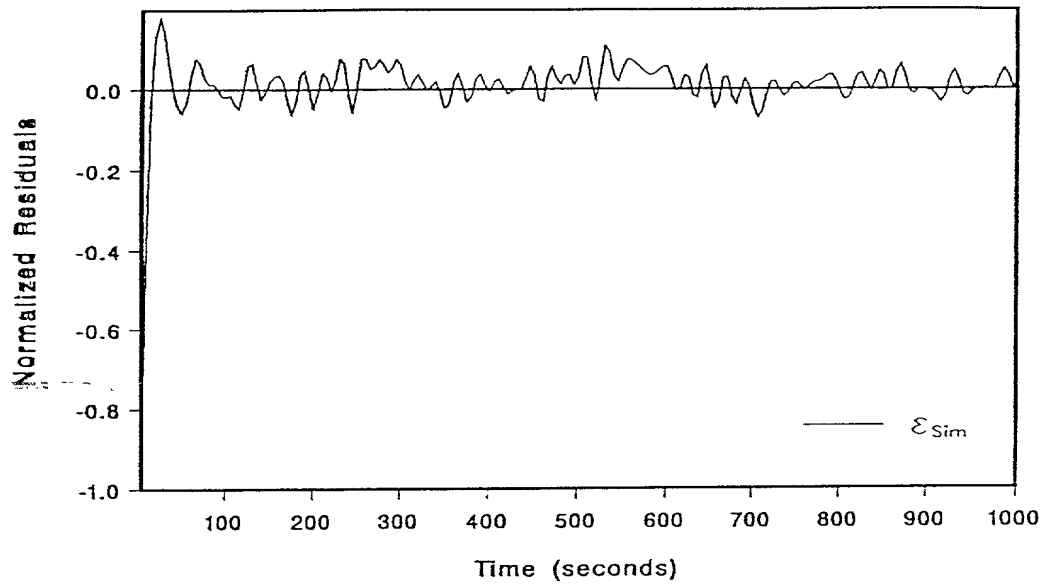


Figure 117. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (Low Noise Environment).

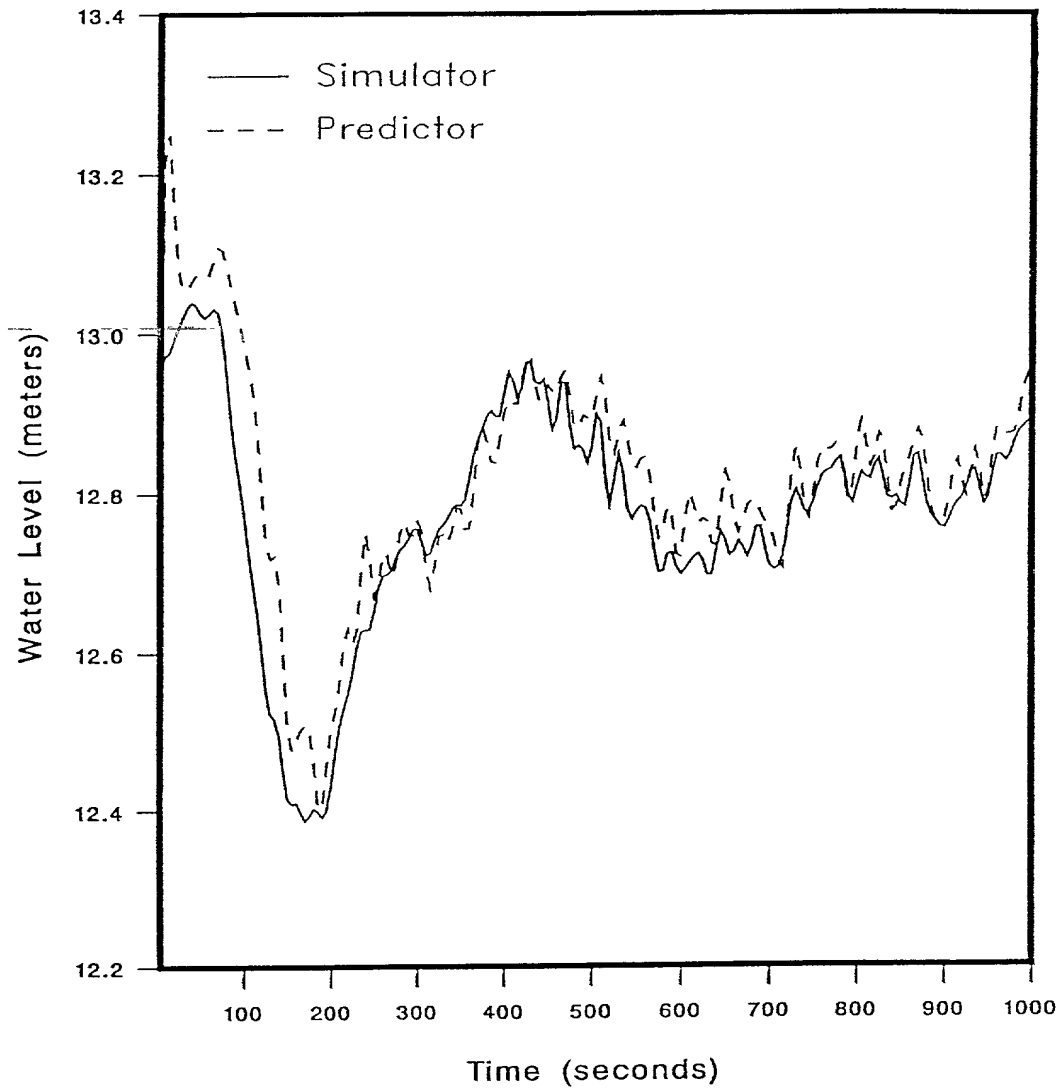


Figure 118. UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (Low Noise Environment).

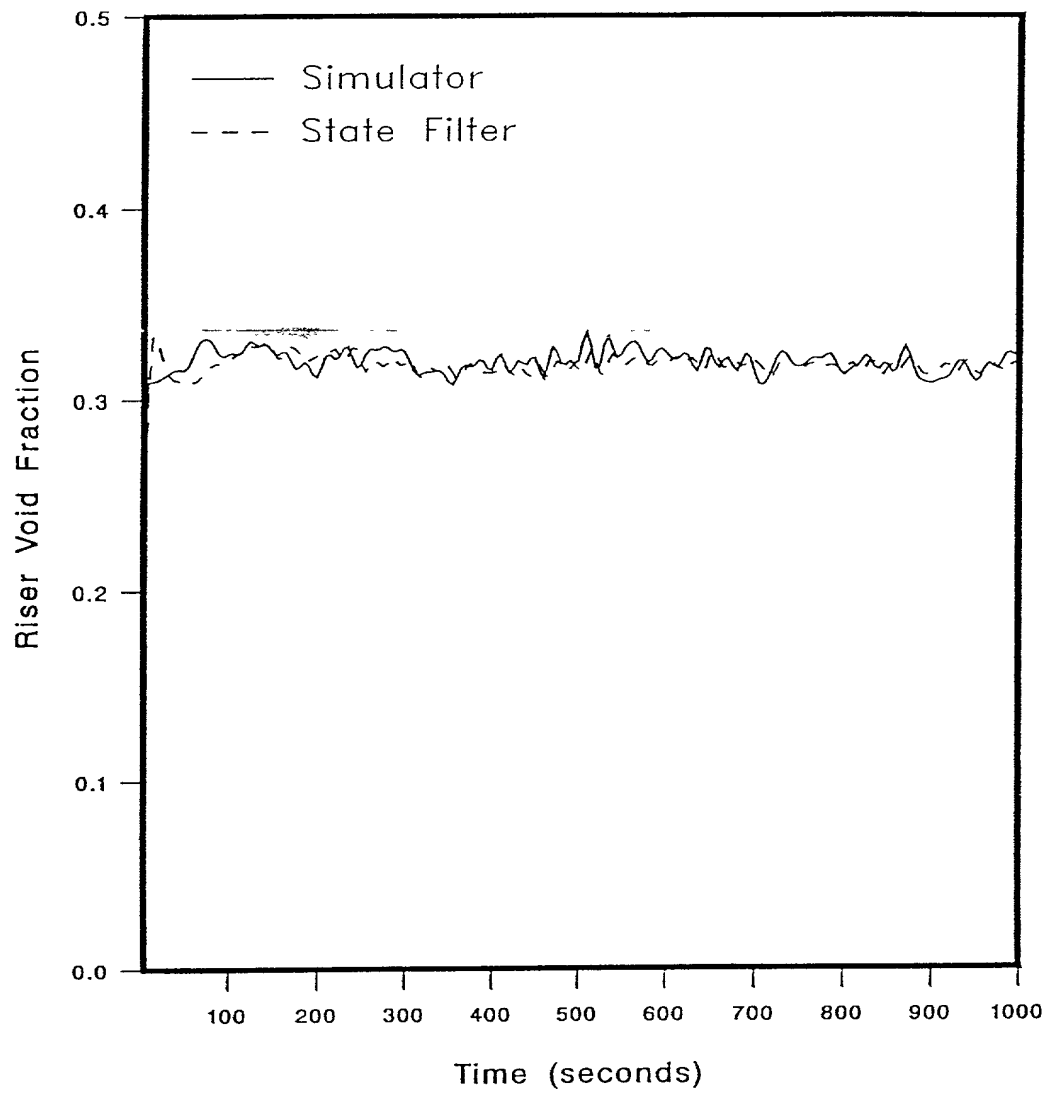


Figure 119. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (Low Noise Environment).

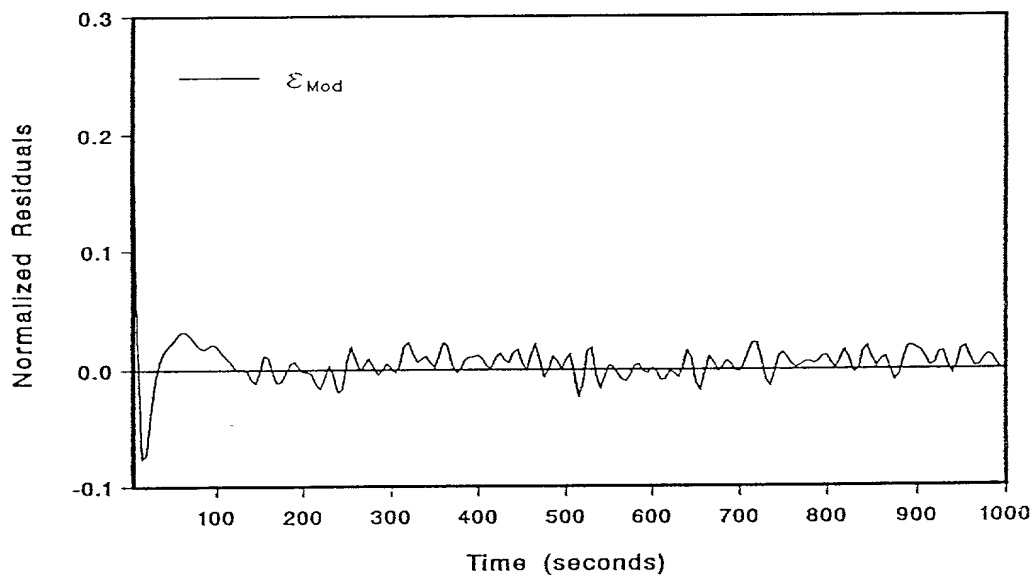
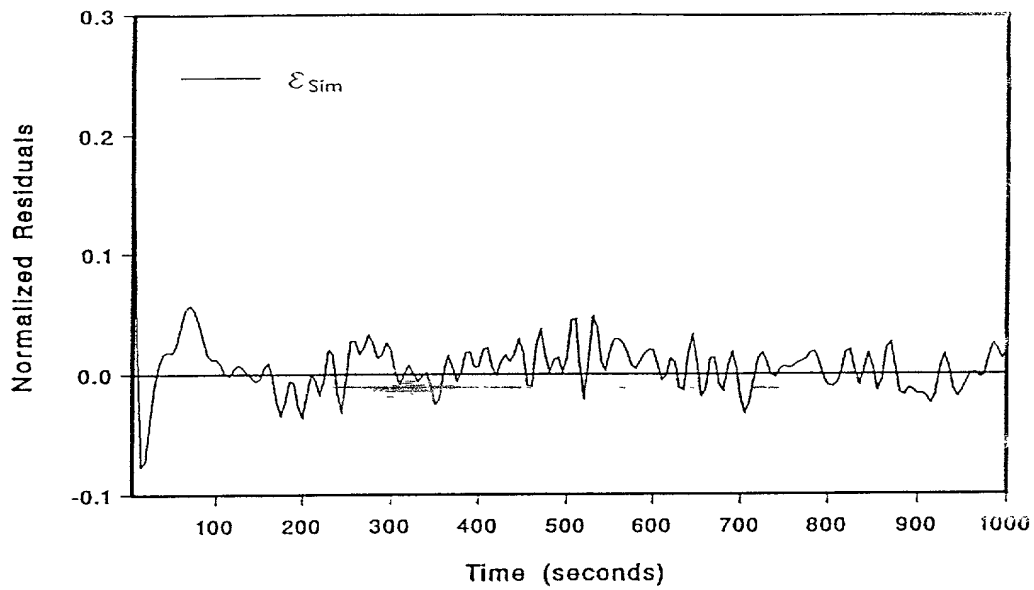


Figure 120. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (Low Noise Environment).

is evaluated. The NN output predictor response to the ramp input is shown in Figure 121. The riser void fraction filtered values, $\hat{\alpha}_{NN,R}(t|t)$, as determined by the NN state filter to the ramp input are shown in Figure 122. The NMSE for the state filtered values is 4.5%. The normalized residuals, $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$, are shown in Figure 123. The average $\epsilon_{Sim}(t)$ value is -2.6% and the the average $\epsilon_{Mod}(t)$ value is -2.48%.

The NN output predictor response to the step input is shown in Figure 124. The riser void fraction filtered values, $\hat{\alpha}_{NN,R}(t|t)$, as determined by the NN state filter are shown in Figure 125. The NMSE for the state filtered values is 0.11%. The normalized residuals, $\epsilon_{Sim}(t)$ and $\epsilon_{Mod}(t)$, are shown in Figure 126. The average $\epsilon_{Sim}(t)$ value is 0.26% and the the average $\epsilon_{Mod}(t)$ value is 0.24%.

It is seen that the Filter 3 state filtered values are about as accurate as those obtained in Filter 2 and much more accurate than the state filtered values obtained in Filter 1. This is because the NNs in Filter 2 and 3 were trained using state information from the UTSG process simulator model. The process simulator model is much more accurate than the UTSG scheduled model which is used to train the NNs in Filter 1.

VII.5.4 Filter 4 - Adaptive Filter for the Primary Heat Transfer Coefficient

The performance of the adaptive state filter developed as Filter 4 on Tests 1 to 8, as listed in Table 9, in the validation data set is determined. Initially, process and measurement noise, zero-mean, white, Gaussian with 0.05 sd (low noise), is added to the validation data set. Tests 1 to 4 are conducted to evaluate the accuracy of the filtered values of the primary heat transfer coefficient, $\hat{U}_{over}(t|t)$, at *steady-state* UTSG operating power levels. The NN state filter value in Tests 1 to 4 and the normalized residuals of the filtered values are shown in Figures 127–130. The NMSE values of the state filter values, $\hat{U}_{NN,over}(t|t)$, for each of the tests, 1 to 4, are 1.2%, 0.8%, 0.5%, and 0.55%, respectively. The average of the normalized residual $\epsilon_{Sim}(t)$ values for each of the tests, 1 to 4, are 4%, 2.5%, 2.7% and 2.8%, respectively.

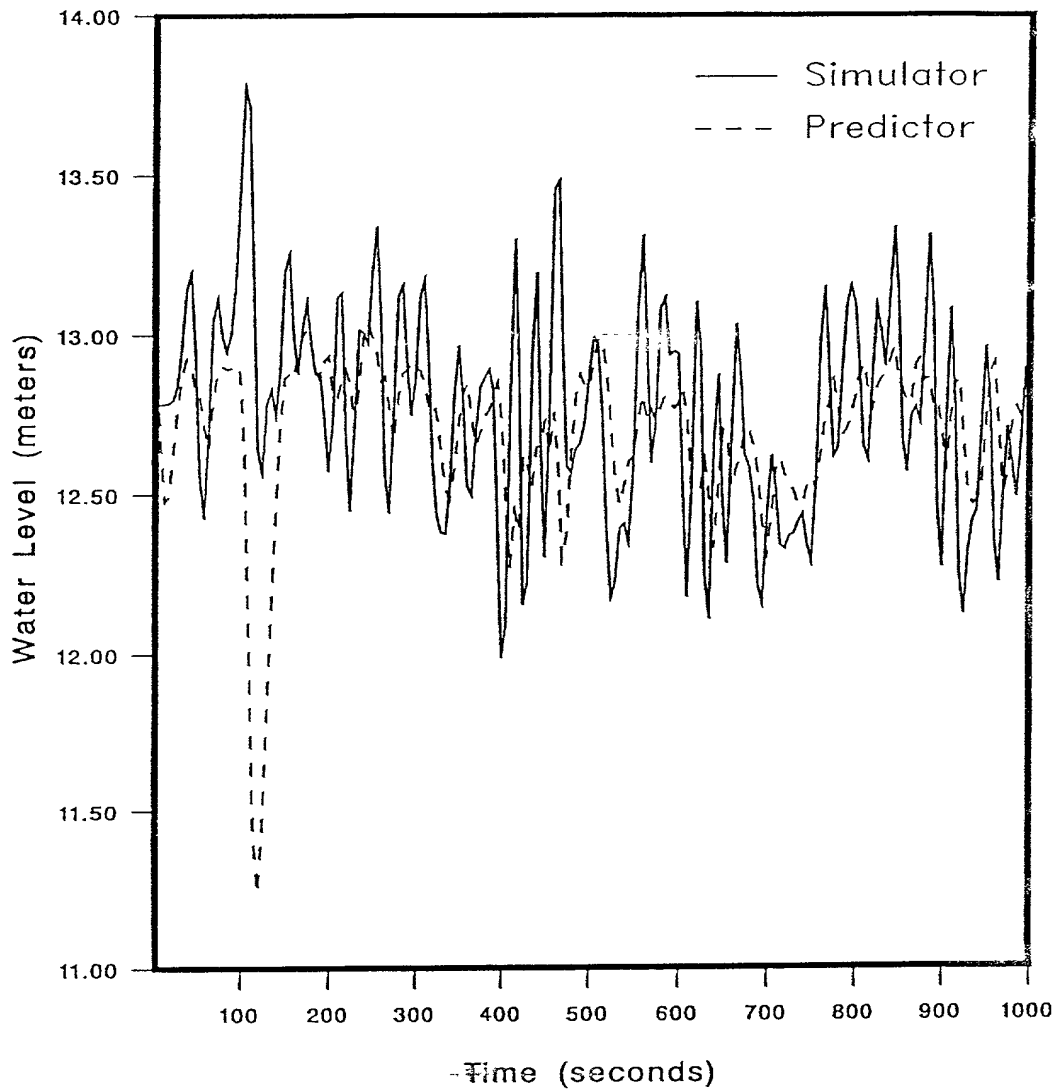


Figure 121. UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (High Noise Environment).

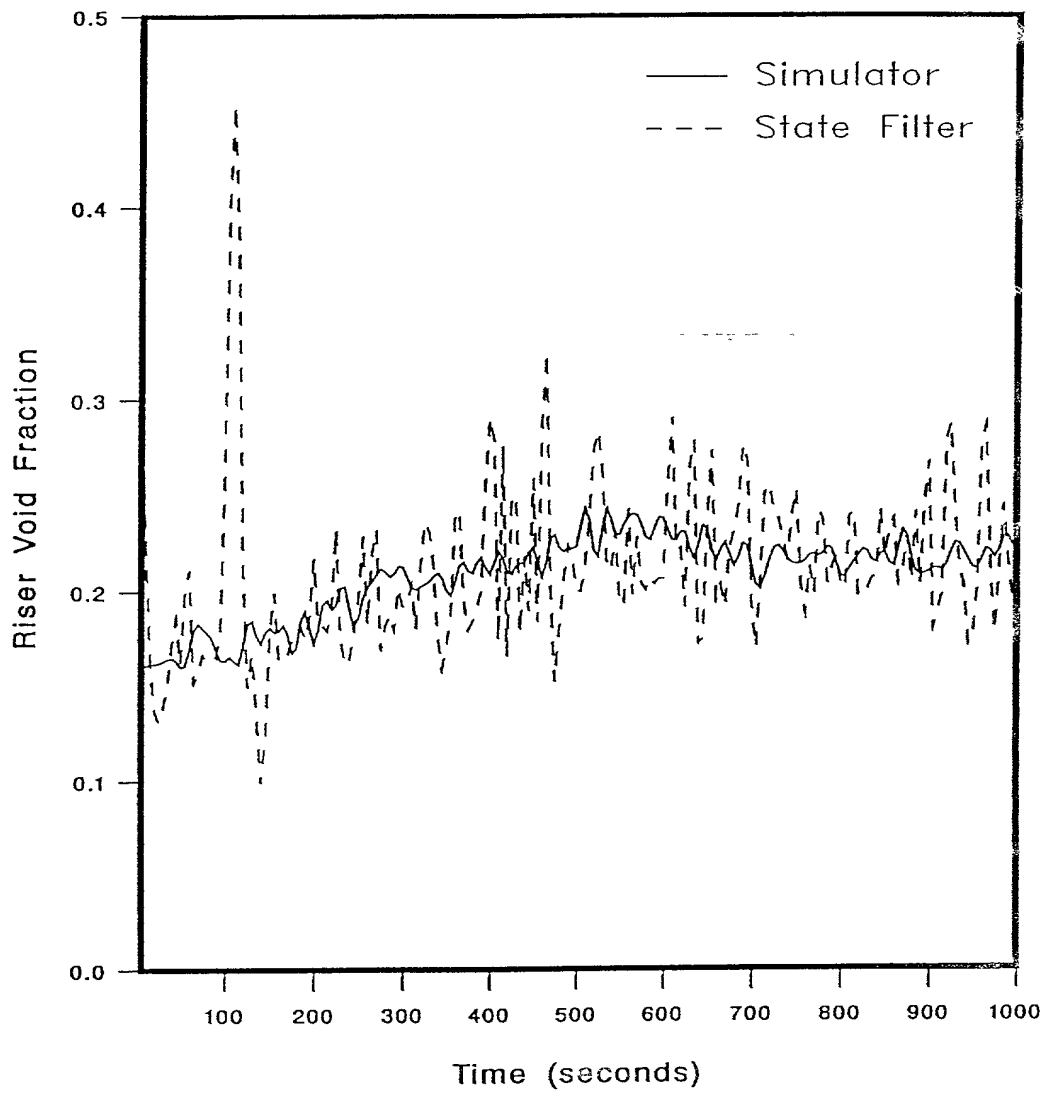


Figure 122. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (High Noise Environment).

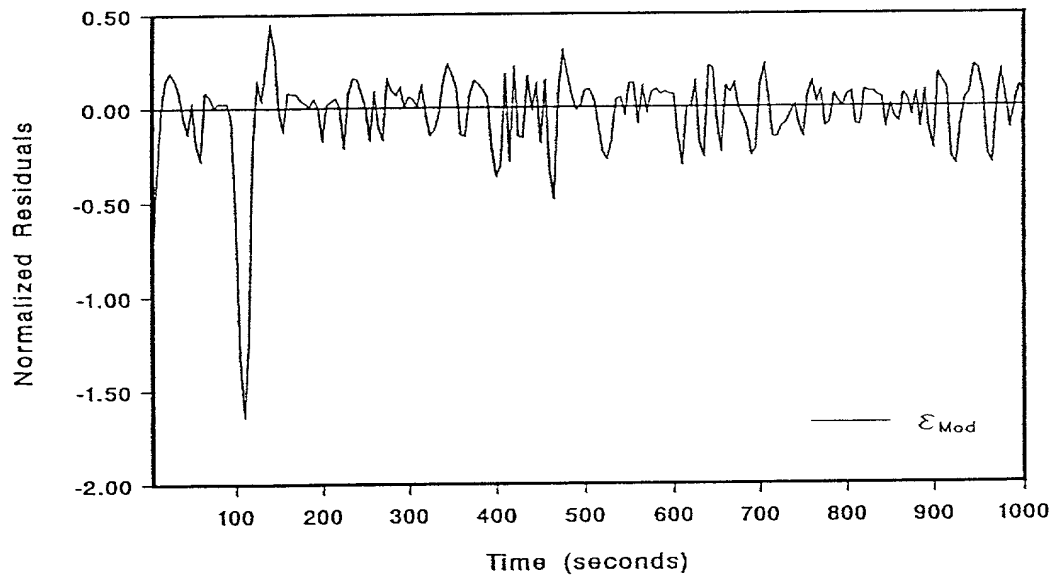
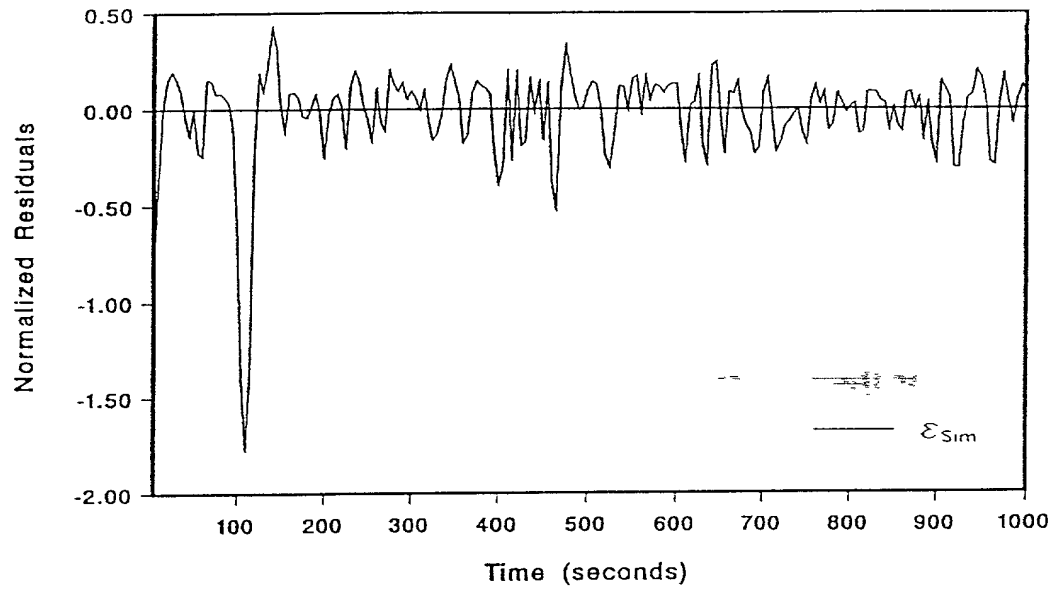


Figure 123. UTSG Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Ramp Input (High Noise Environment).

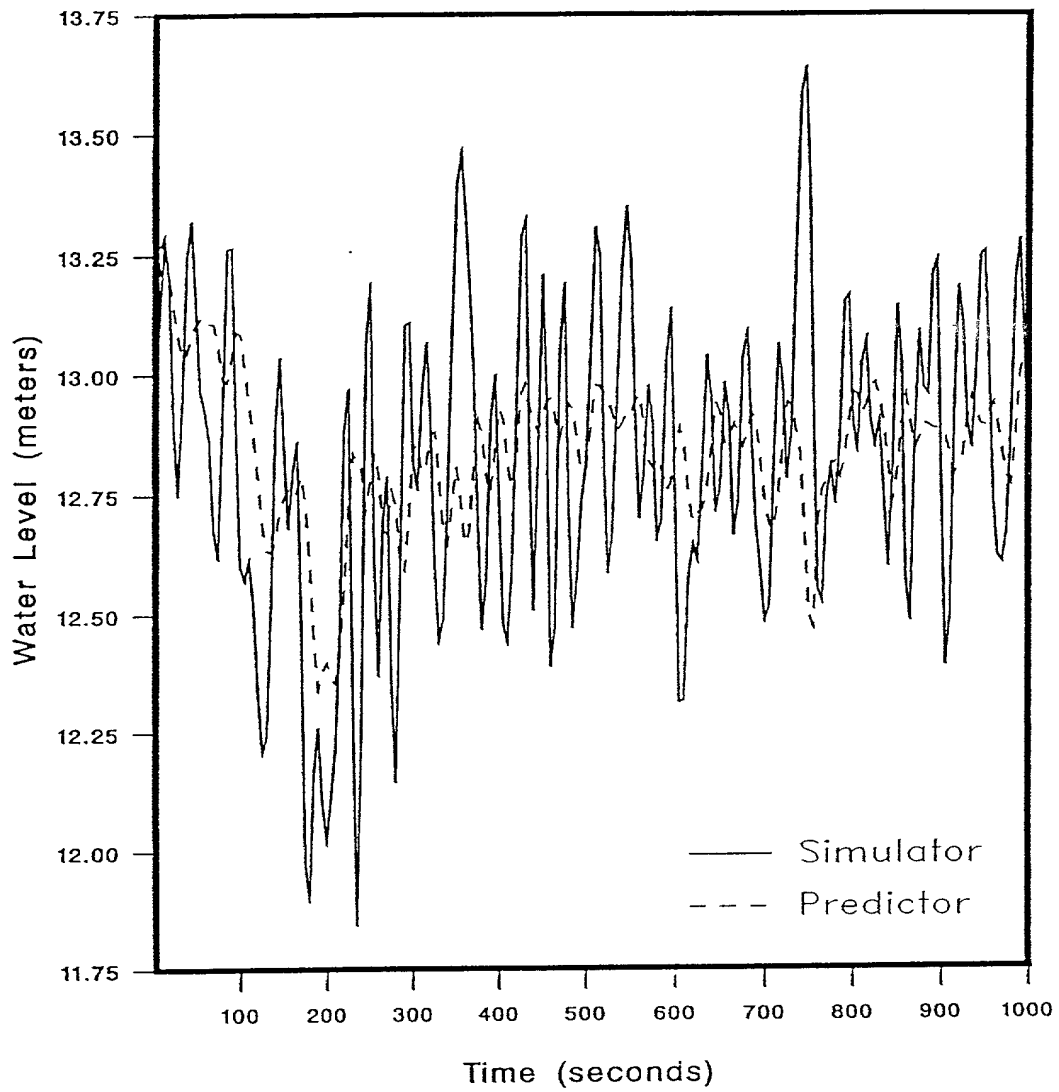


Figure 124. UTSG Process Water Level Response, from the Simulator and NN Output Predictor, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (High Noise Environment).

600

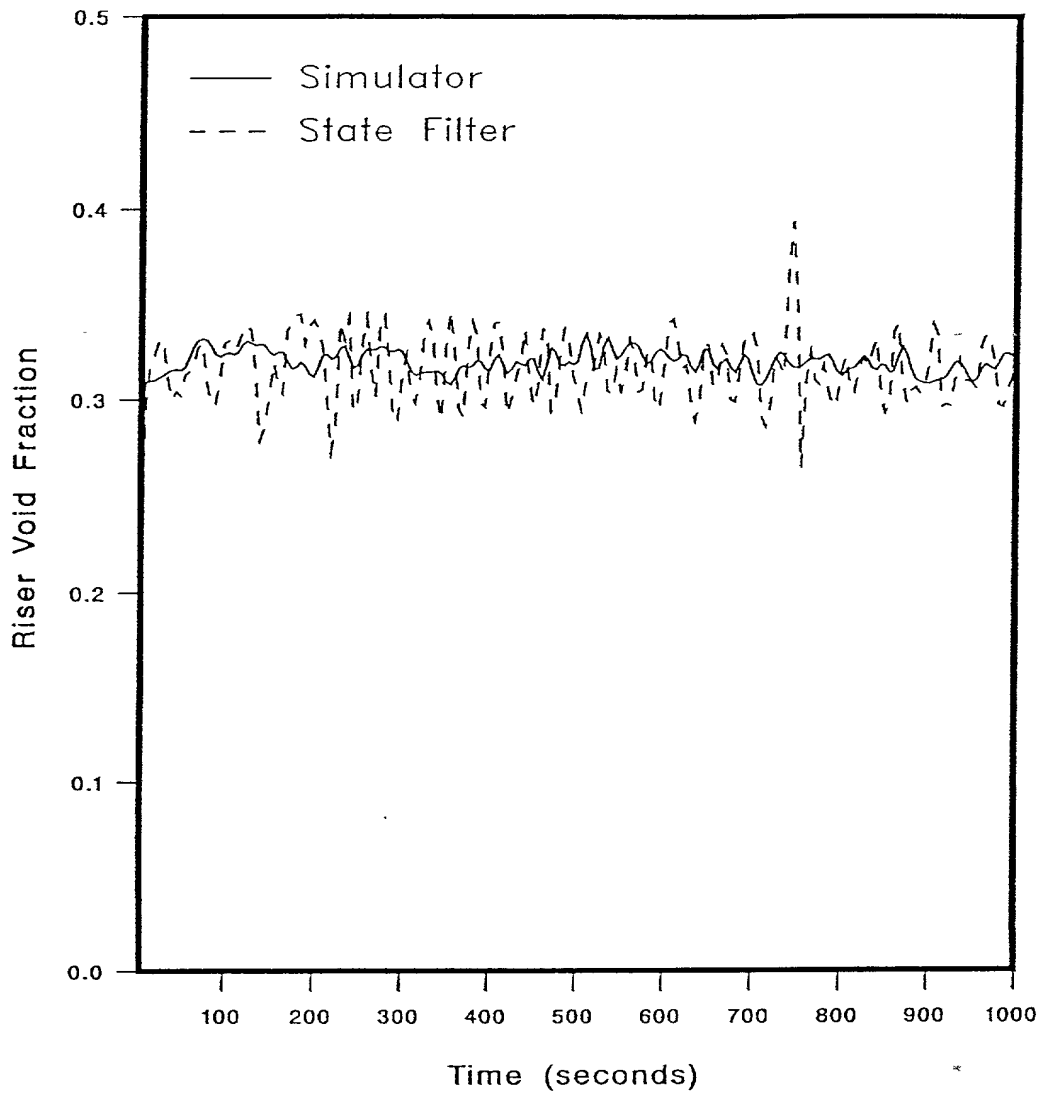


Figure 125. UTSG Process Riser Void Fraction Response, from the Simulator and NN State Filter, for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (High Noise Environment).

601

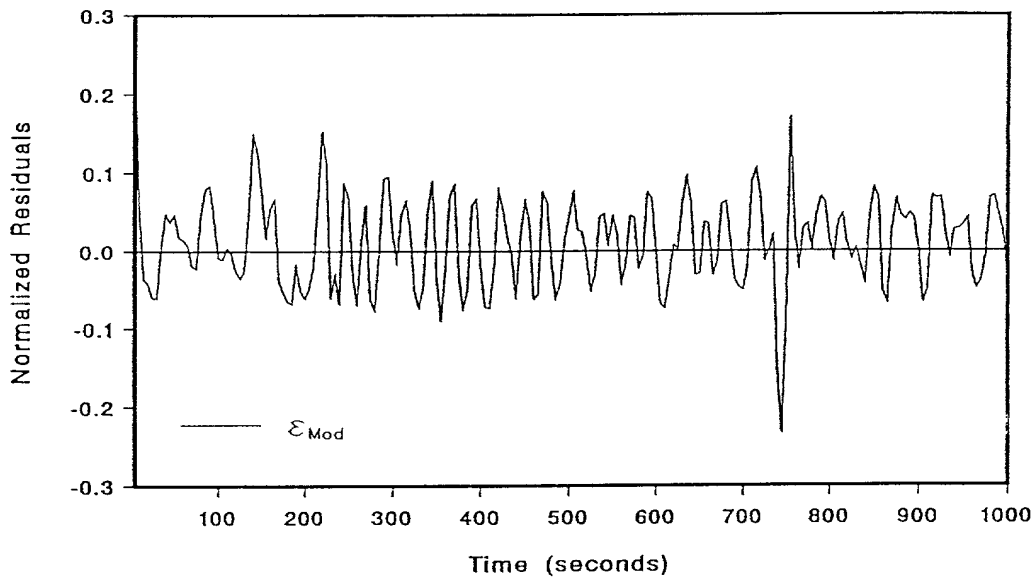
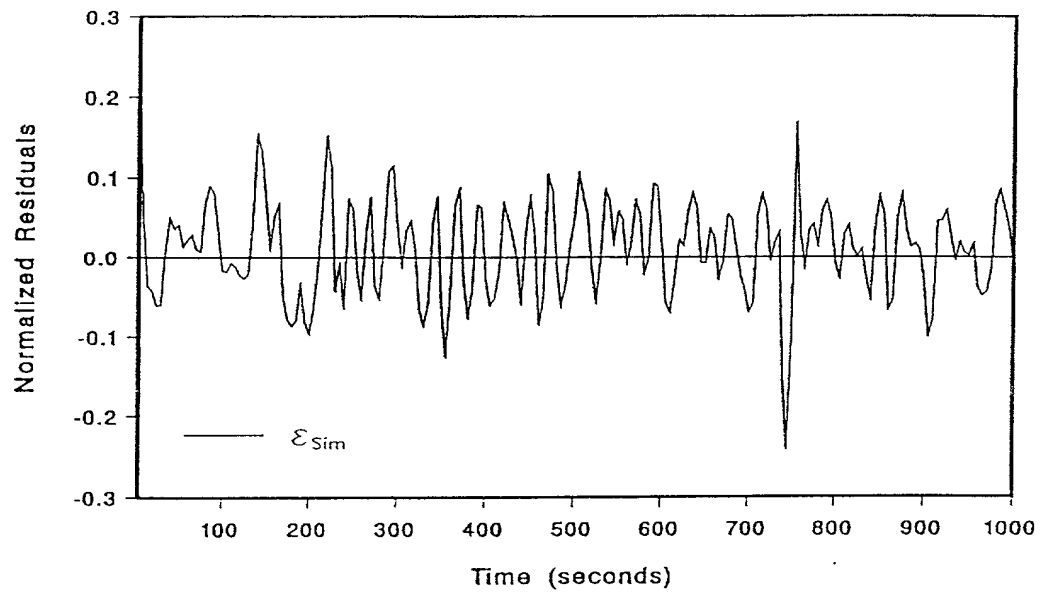


Figure 126. UTSG Process Riser Void Fraction Filter Normalized Residuals for the Adaptive NN Riser Void Fraction Filter 3 Using a Step Input (High Noise Environment).

602

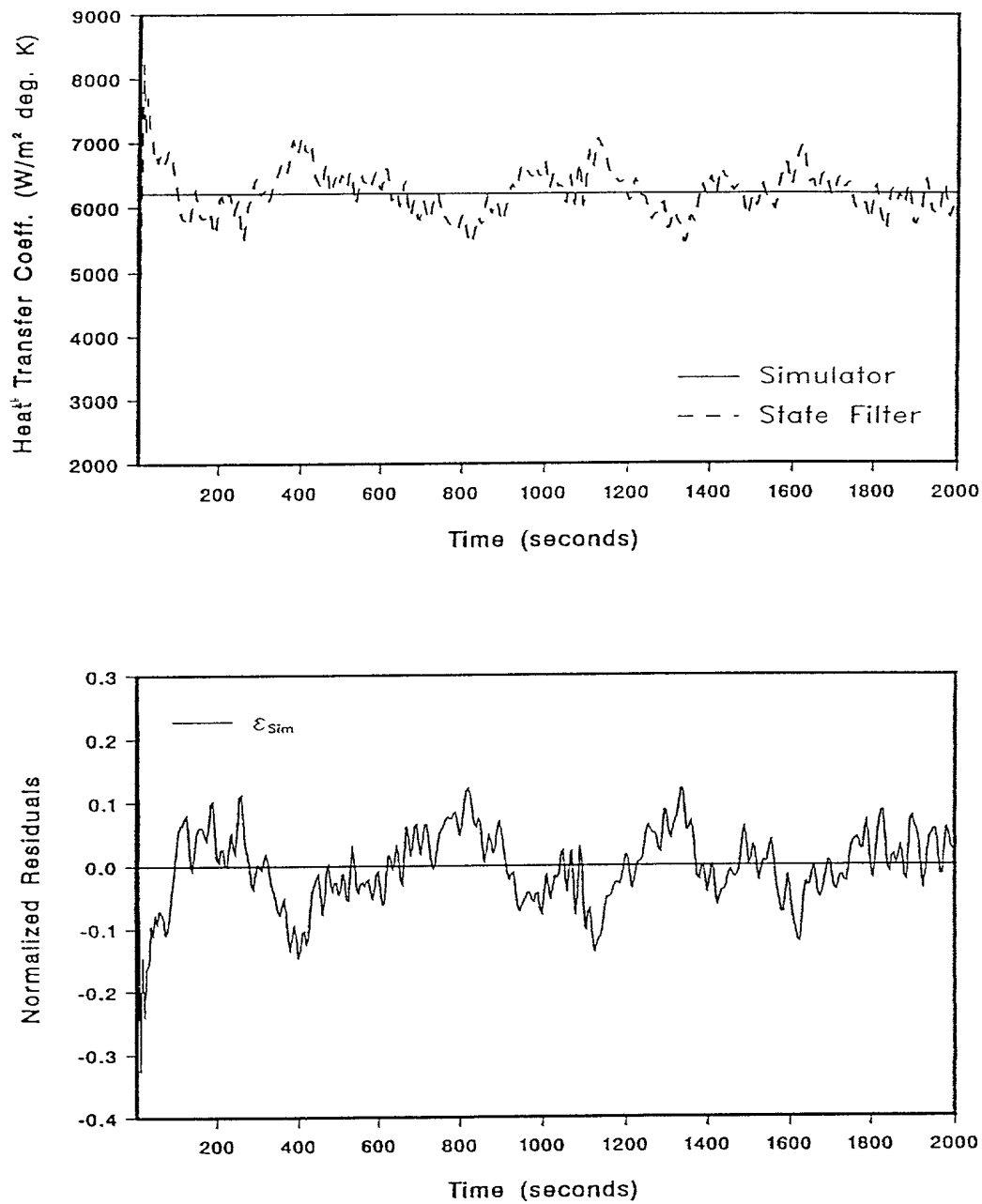


Figure 127. UTSG Process HTC, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 1 (Steady-State at 5% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

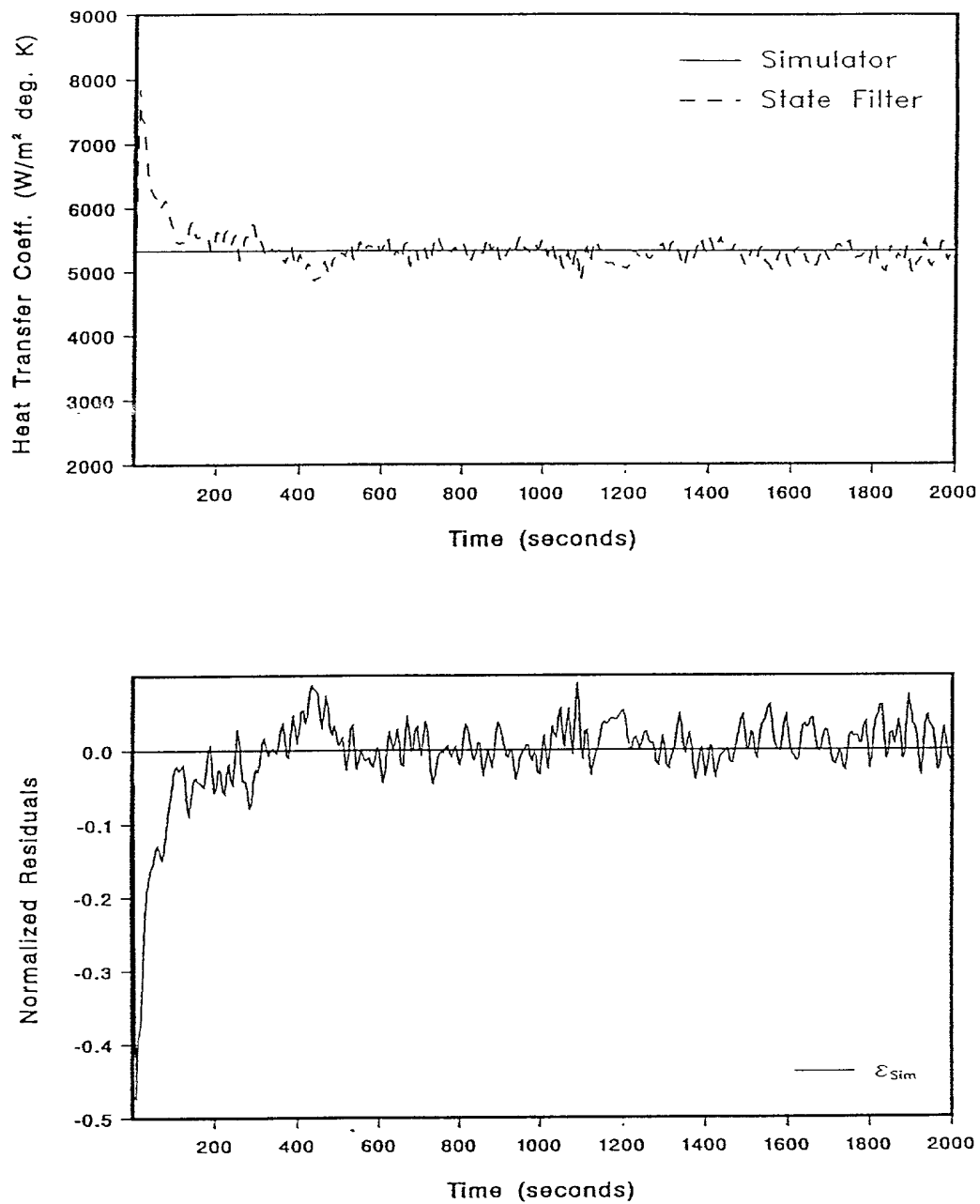


Figure 128. UTSG Process HTD Values, from the Simulator and NN State Filter, for the Adaptive NN HTD Filter 4 Using Test 2 (Steady-State at 10% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

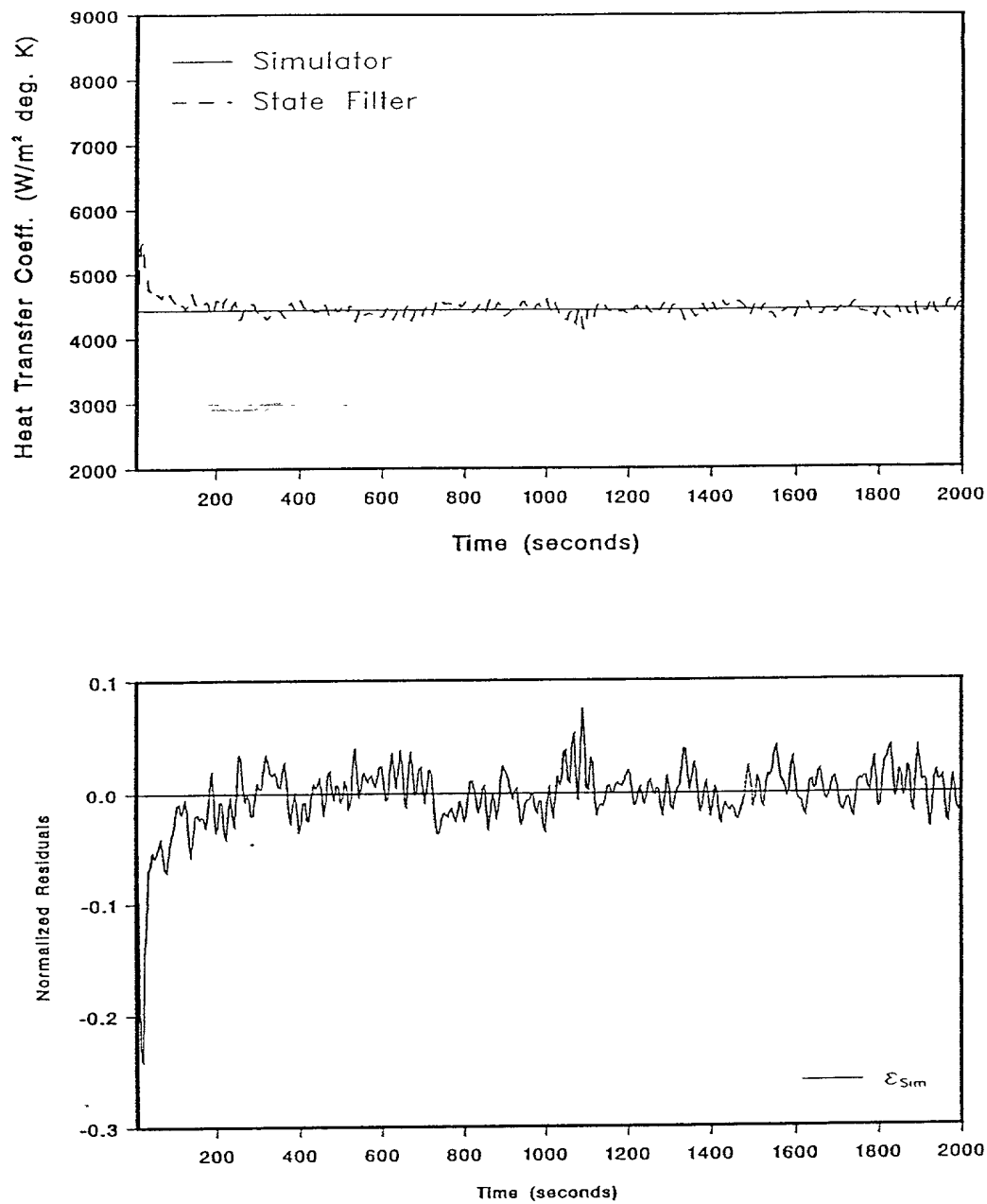


Figure 129. UTSG Process HTc Values, from the Simulator and NN State Filter, for the Adaptive NN HTc Filter 4 Using Test 3 (Steady-State at 15% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

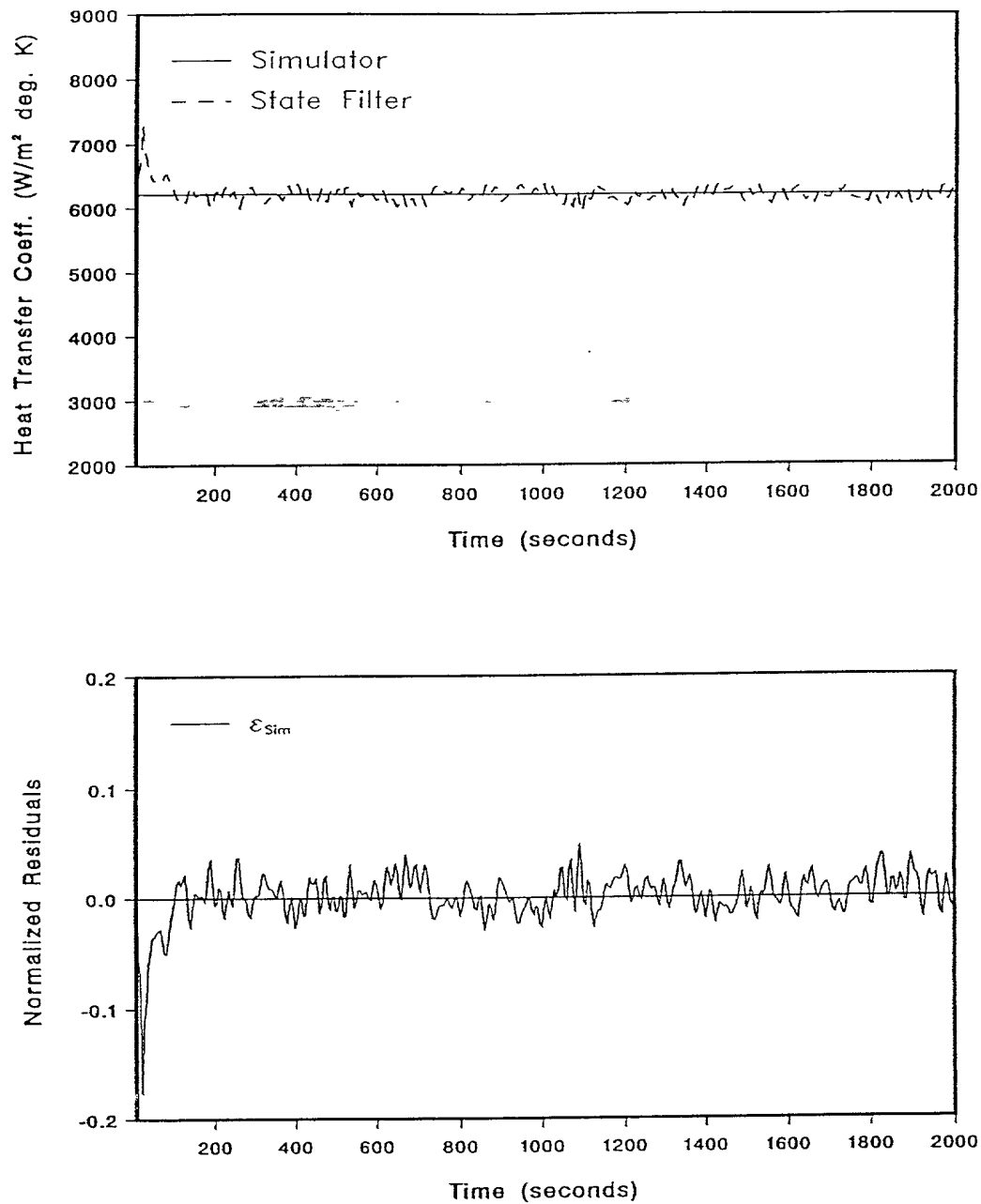


Figure 130. UTSG Process HTc Values, from the Simulator and NN State Filter, for the Adaptive NN HTc Filter 4 Using Test 4 (Steady-State at 20% of Full Operating Power and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

Tests 5 to 8 are conducted at ramped or step increases in the UTSG operating power level. The UTSG NN output predictor transient response in Test 5 is shown in Figure 131. The NN state filter response, and the corresponding $\epsilon_{\text{Sim}}(t)$ values in Test 5 are shown in Figure 132. The NMSE for the state filtered values is 0.46% and the average $\epsilon_{\text{Sim}}(t)$ value is 1.31%. The UTSG NN output predictor transient response in Test 6 is shown in Figure 133. The state filter values and the corresponding $\epsilon_{\text{Sim}}(t)$ values in Test 6 are shown in Figure 134. The NMSE for the state filter values is 0.58% and the average $\epsilon_{\text{Sim}}(t)$ value is 0.73%. The NN output predictor transient response in Test 7 is shown in Figure 135. The state filter values and the corresponding $\epsilon_{\text{Sim}}(t)$ values in Test 7 are shown in Figure 136. The NMSE for the state filter values is 0.31% and the average $\epsilon_{\text{Sim}}(t)$ value is -0.85%. The NN output predictor transient response in Test 8 is shown in Figure 137. The state filter values and the corresponding $\epsilon_{\text{Sim}}(t)$ values in Test 8 are shown in Figure 138. The NMSE for the state filter values is 2.9% and the average $\epsilon_{\text{Sim}}(t)$ value is -0.32%.

The NN output predictor response in Test 5 with high process and measurement noise (zero-mean, white, Gaussian and 0.15 sd) is shown in Figure 139. The state filter values and the corresponding $\epsilon_{\text{Sim}}(t)$ values in Test 5 are shown in Figure 140. The NMSE for the state filter values is 0.6% and the average $\epsilon_{\text{Sim}}(t)$ value is 1.11%. The NN output predictor response in Test 6 with high process and measurement noise is shown in Figure 141. The state filter values and the corresponding $\epsilon_{\text{Sim}}(t)$ values in Test 6 are shown in Figure 142. The NMSE for the state filter values is 1.28% and the average $\epsilon_{\text{Sim}}(t)$ value is -2.906%.

VII.6 Chapter Summary

A UTSG process system is used to develop state filters based on NN state filtering methods developed in Chapter IV. The UTSG system is highly complex system and is an important component of nuclear power plants. The UTSG simulator used in this study is one that has previously been validated with actual plant data.

607

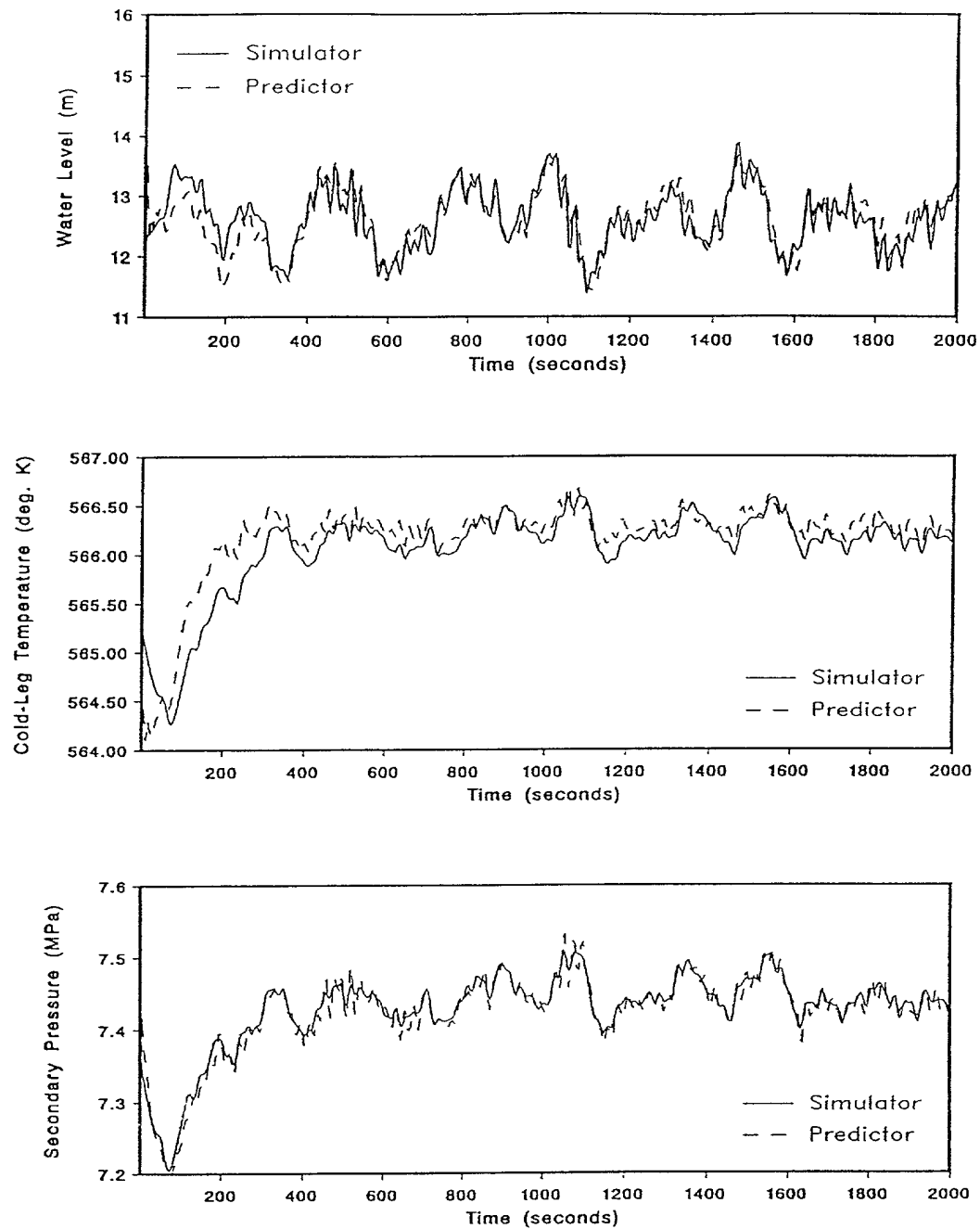


Figure 131. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and Low Noise Environment).

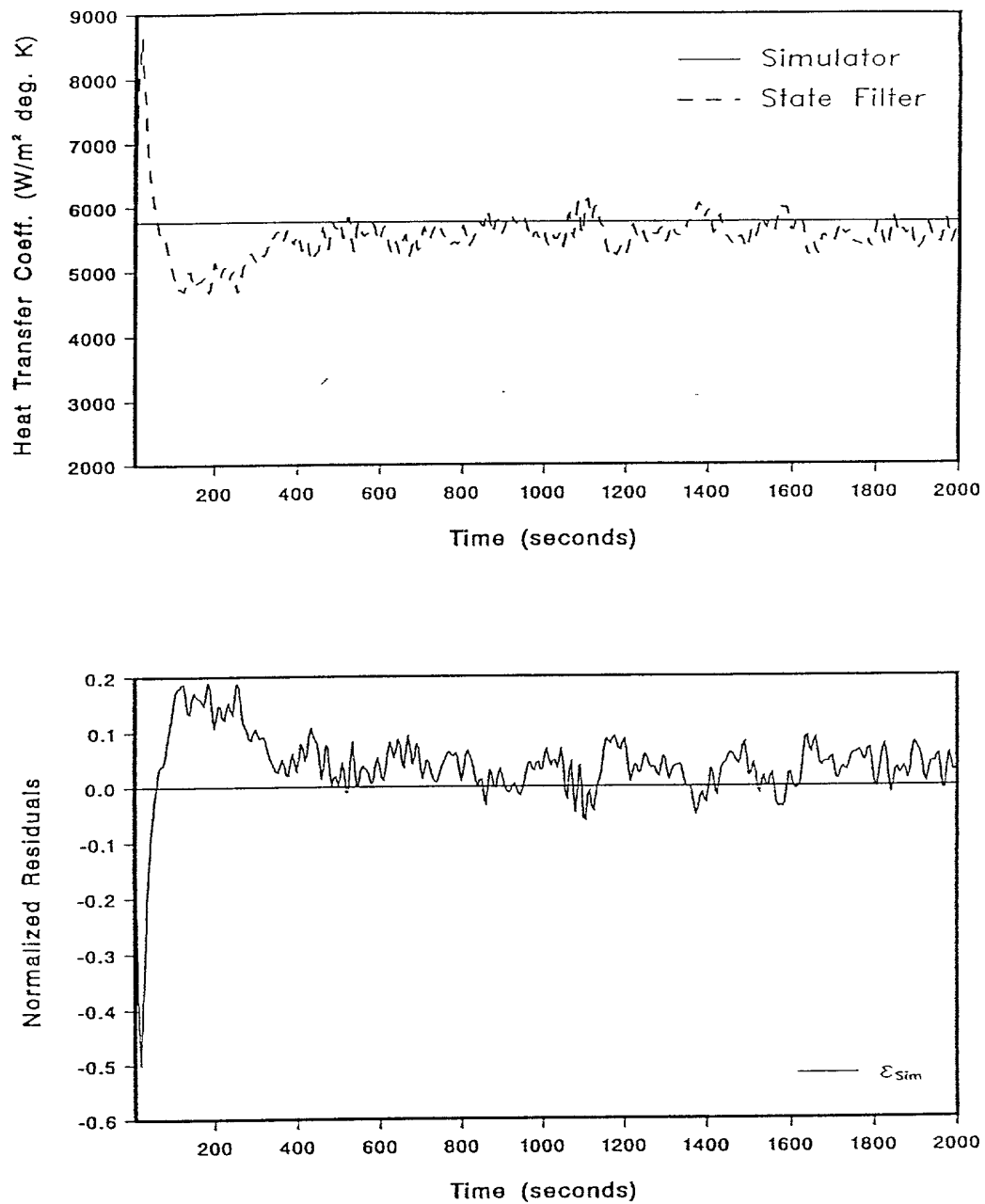


Figure 132. UTSG Process HTc Values, from the Simulator and NN State Filter, for the Adaptive NN HTc Filter 4 Using Test 5 (Ramp Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

609

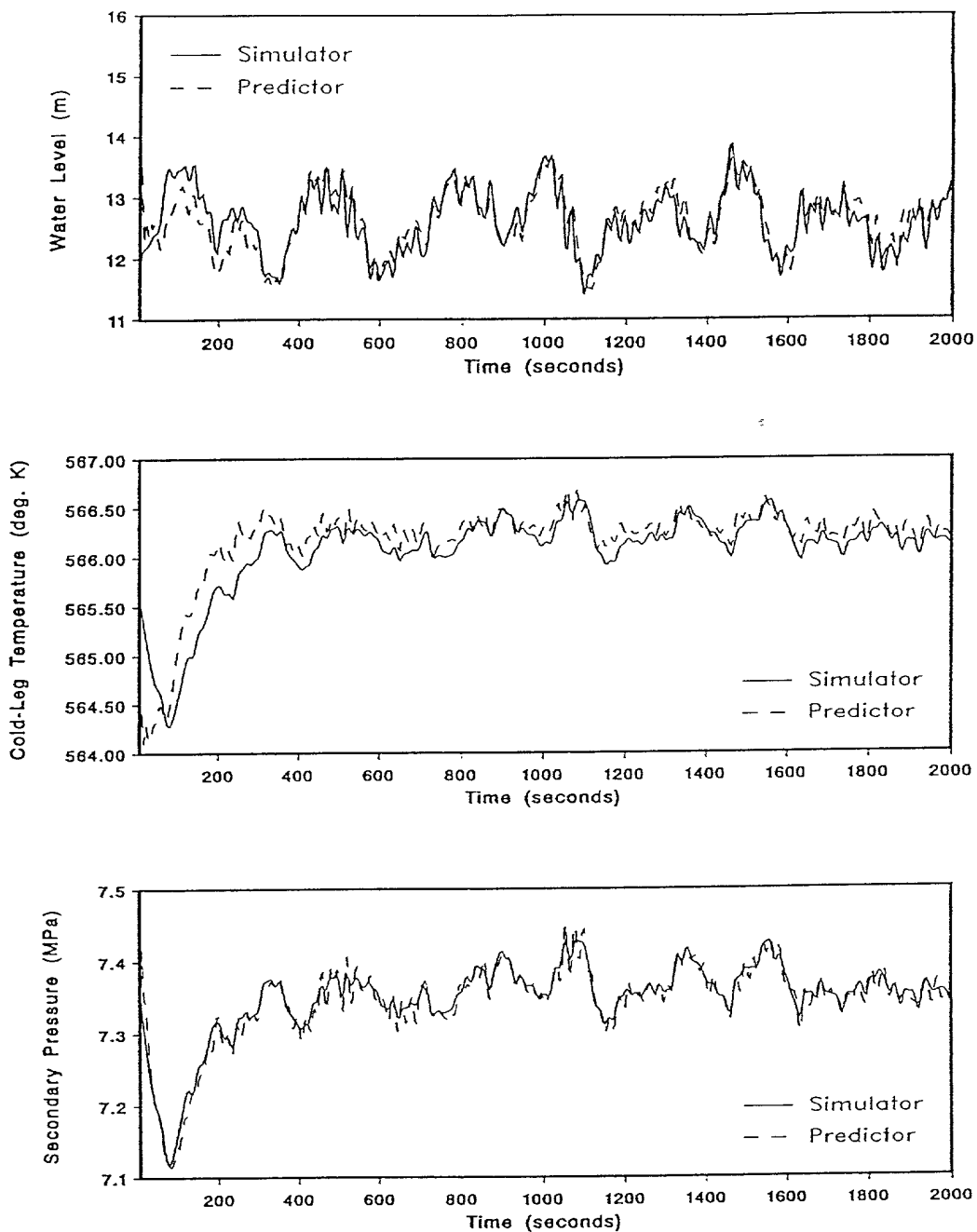


Figure 133. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 6 (Step Input and Low Noise Environment).

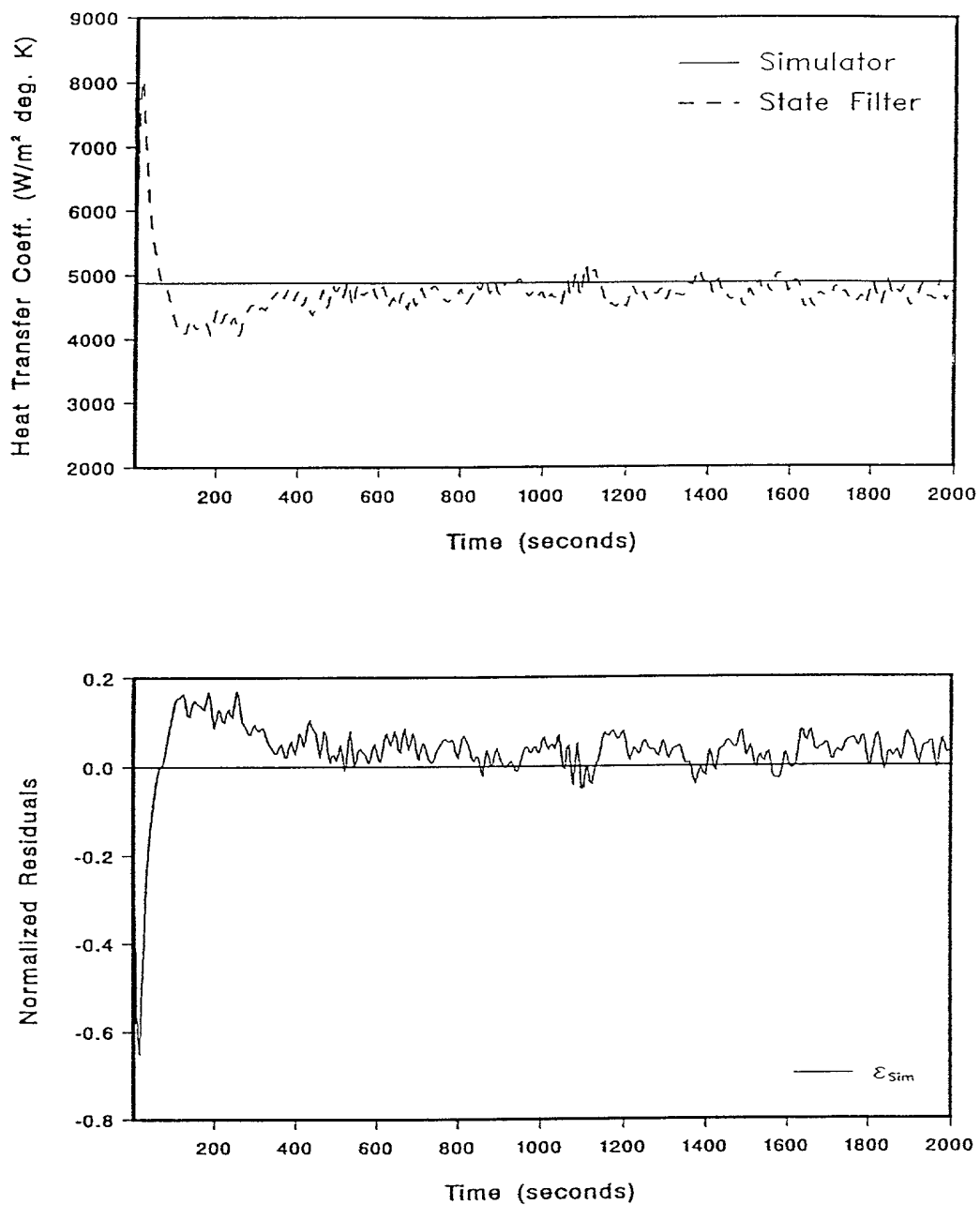


Figure 134. UTSG Process HTc Values, from the Simulator and NN State Filter, for the Adaptive NN HTc Filter 4 Using Test 6 (Step Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

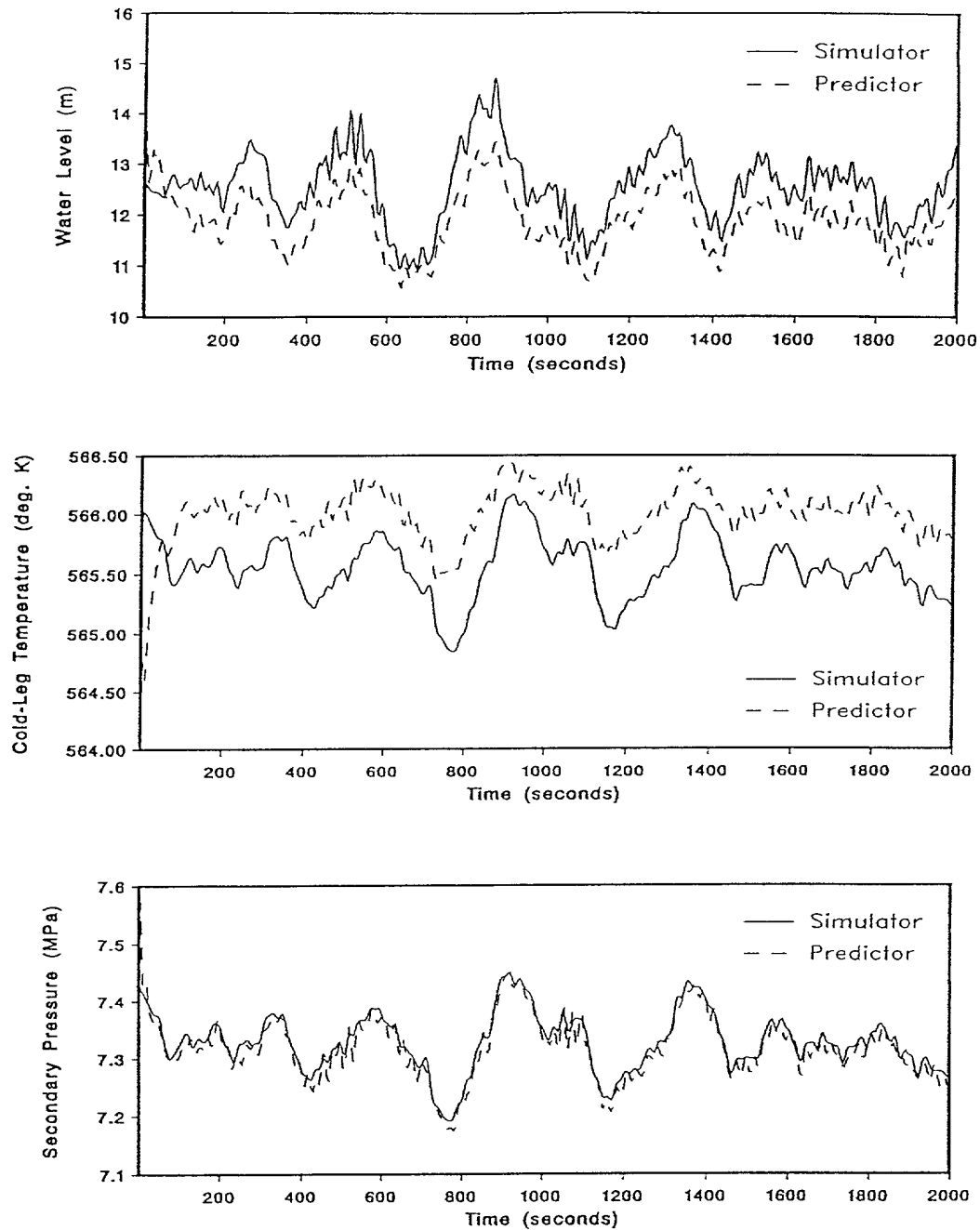


Figure 135. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 7 (Ramp Input and Low Noise Environment).

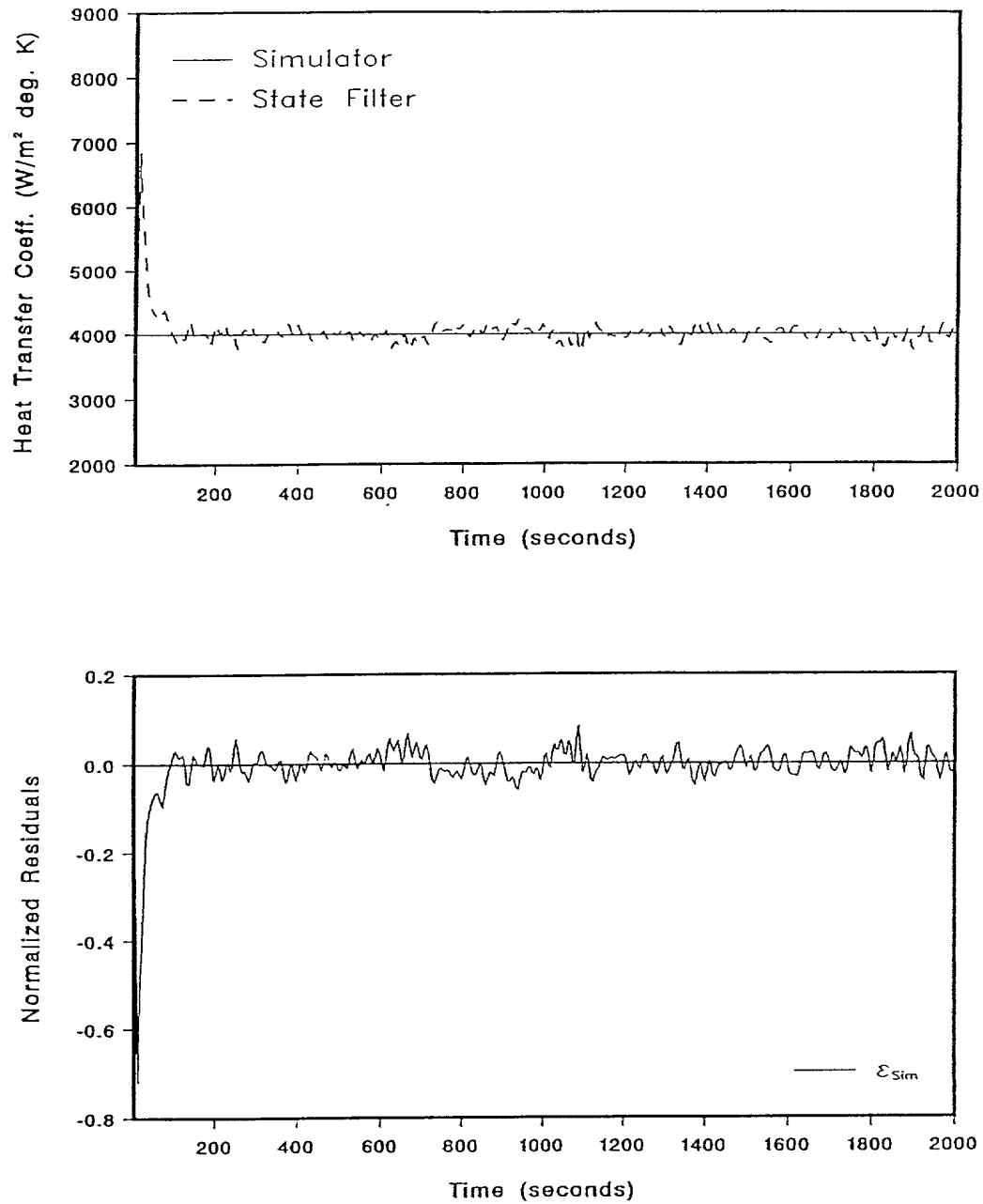


Figure 136. UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 7 (Ramp Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

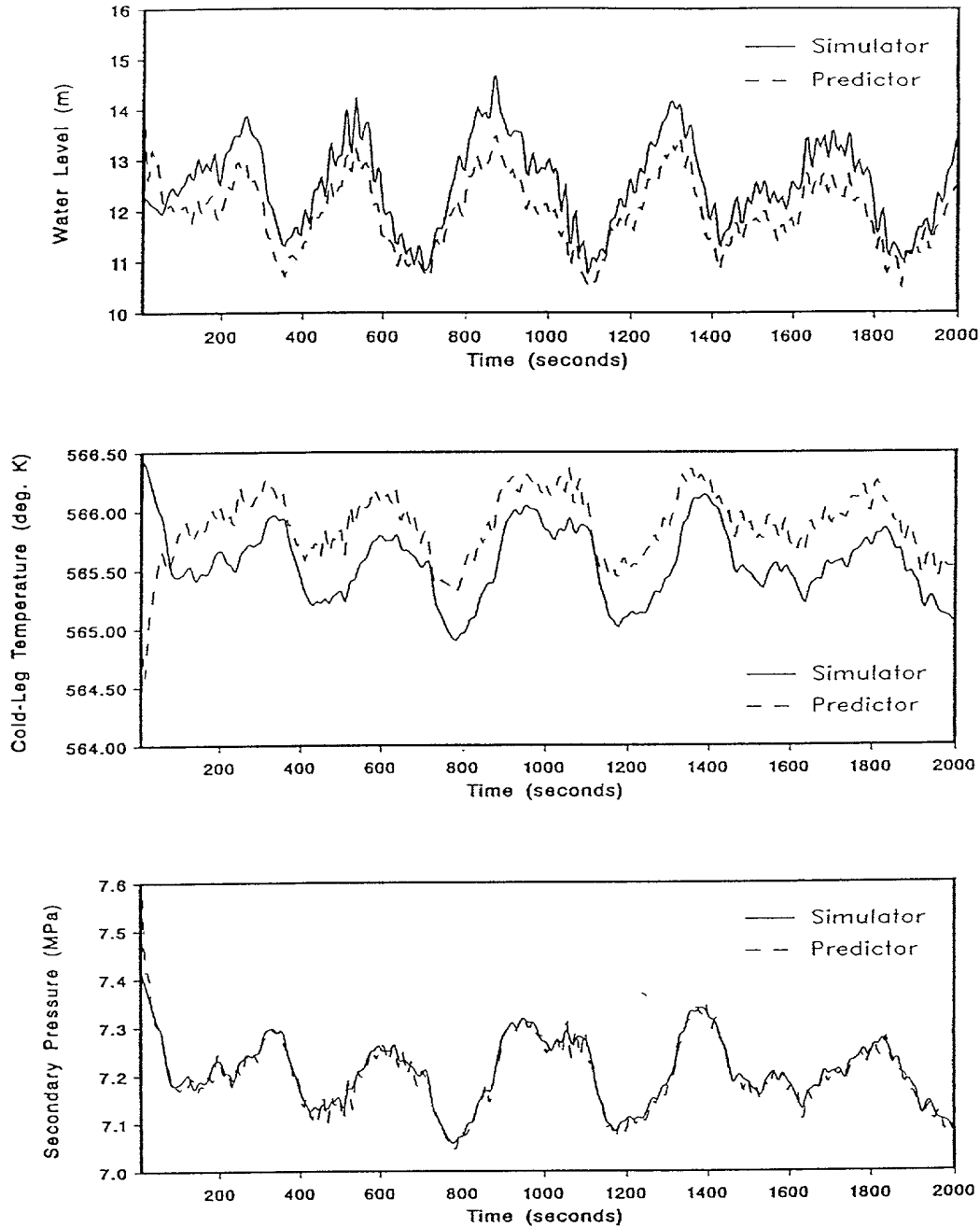


Figure 137. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 8 (Step Input and Low Noise Environment).

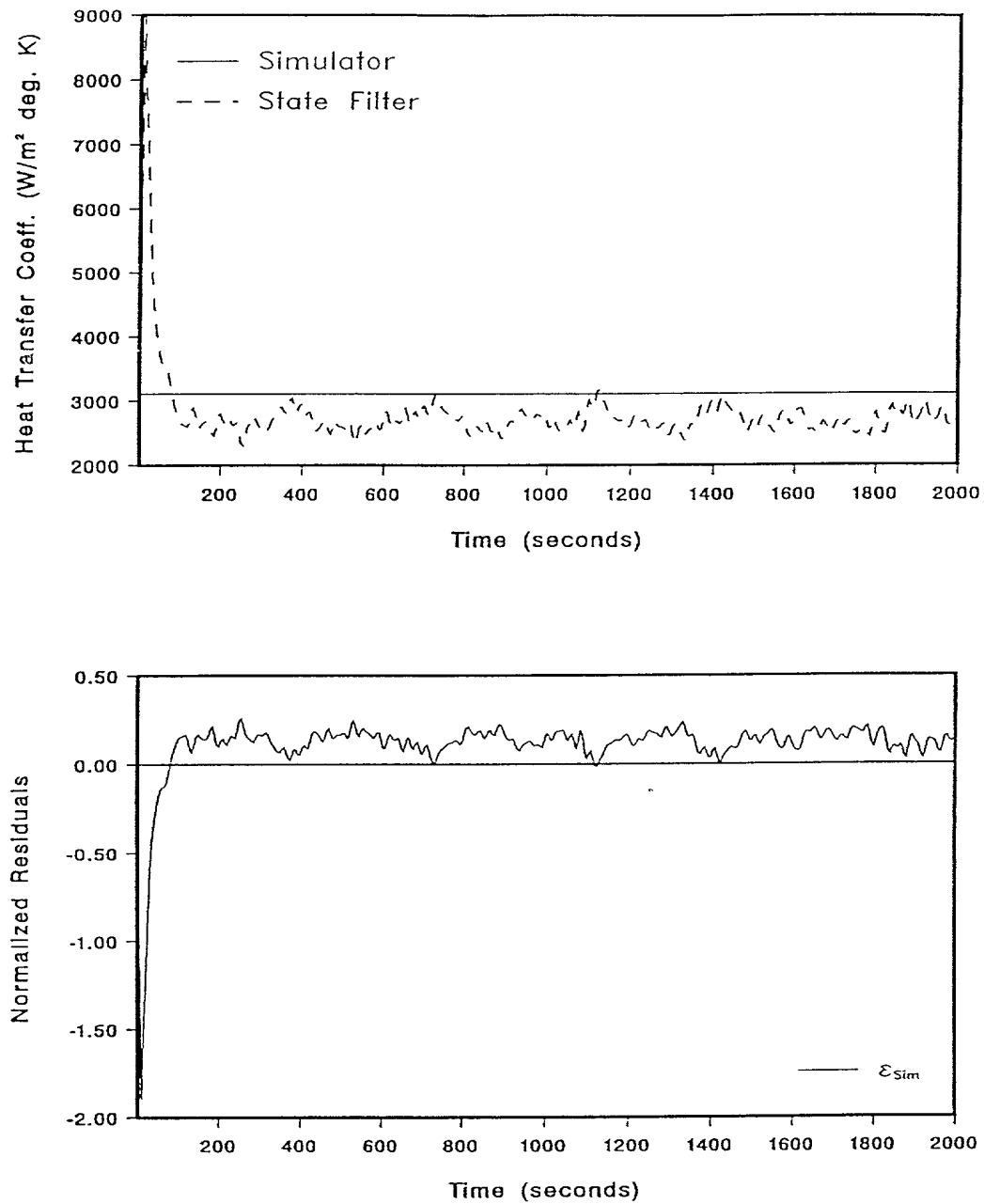


Figure 138. UTSG Process HTc Values, from the Simulator and NN State Filter, for the Adaptive NN HTc Filter 4 Using Test 8 (Step Input and Low Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

615

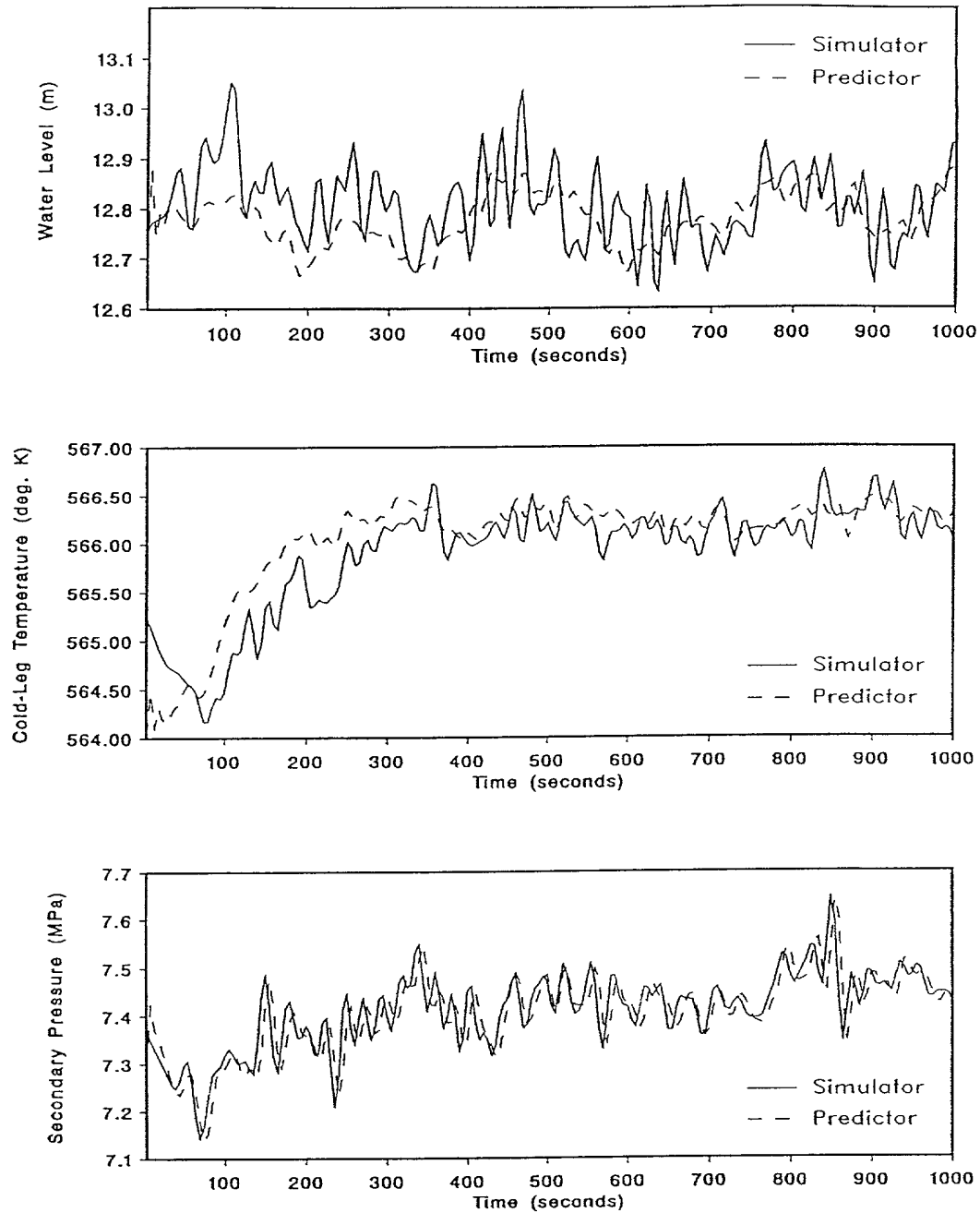


Figure 139. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and High Noise Environment).

666

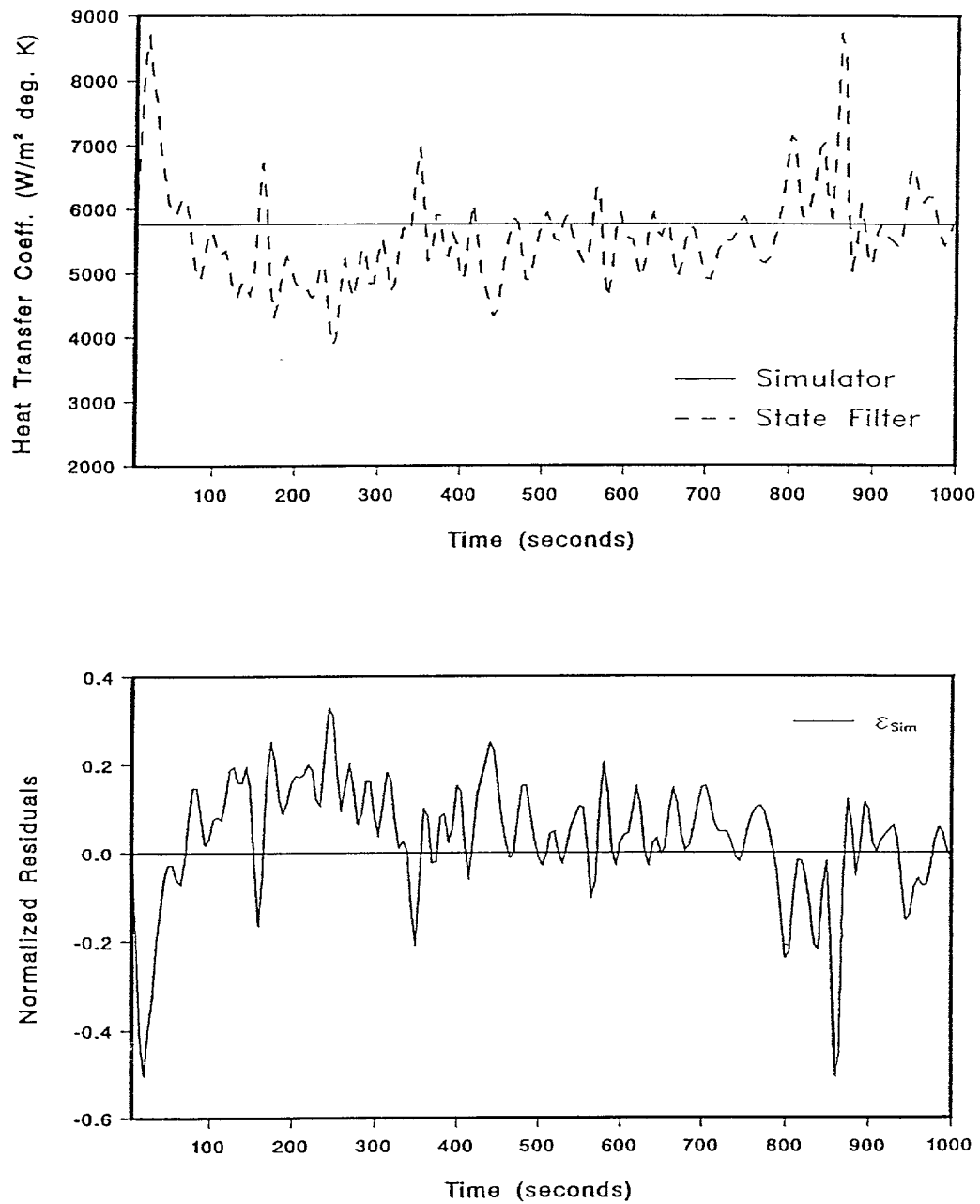


Figure 140. UTSG Process HTC Values, from the Simulator and NN State Filter, for the Adaptive NN HTC Filter 4 Using Test 5 (Ramp Input and High Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

617

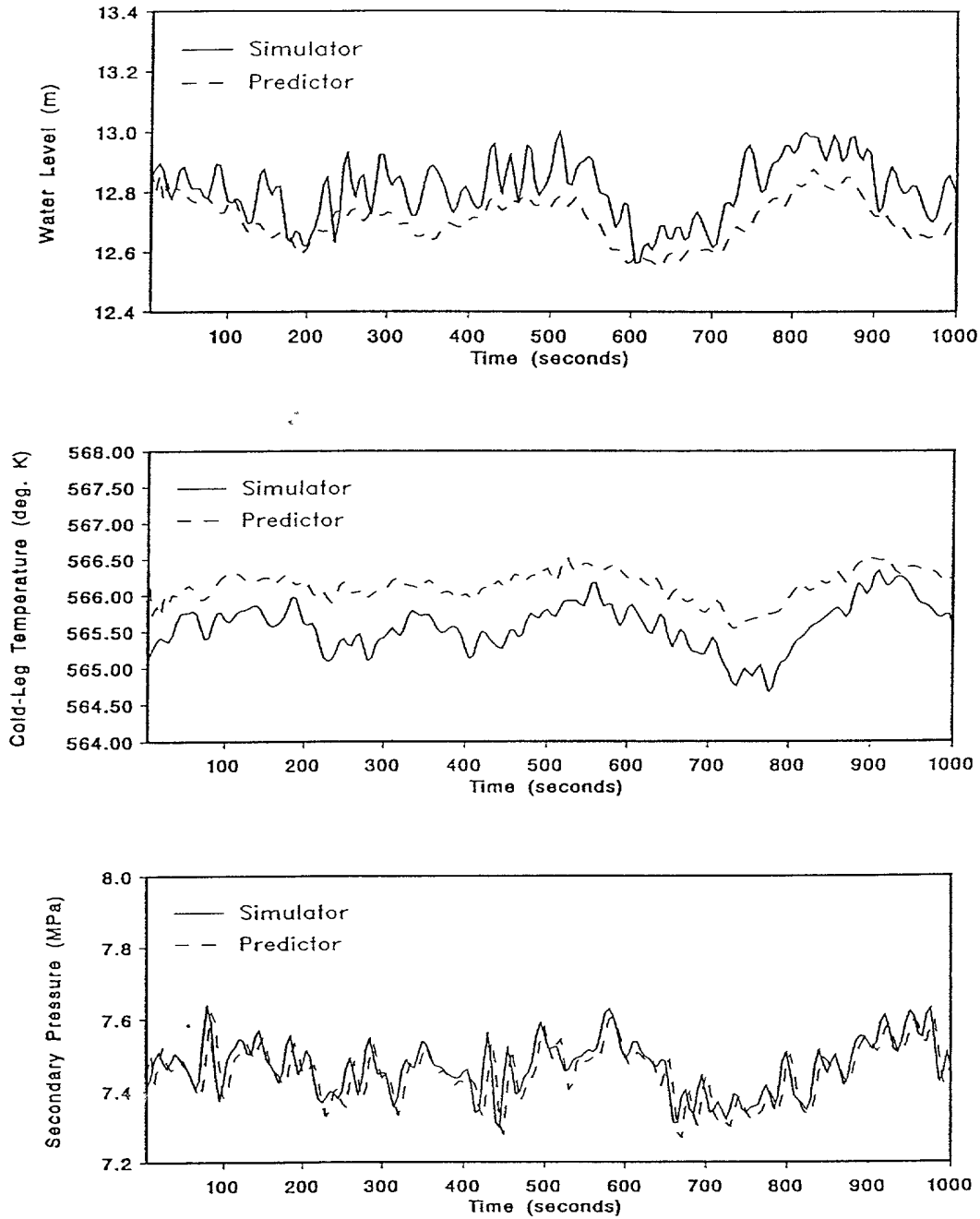


Figure 141. UTSG Process Output Response, from the Simulator and NN Output Predictor, for the Adaptive NN HTC Filter 4 Using Test 6 (Step Input and High Noise Environment).

618

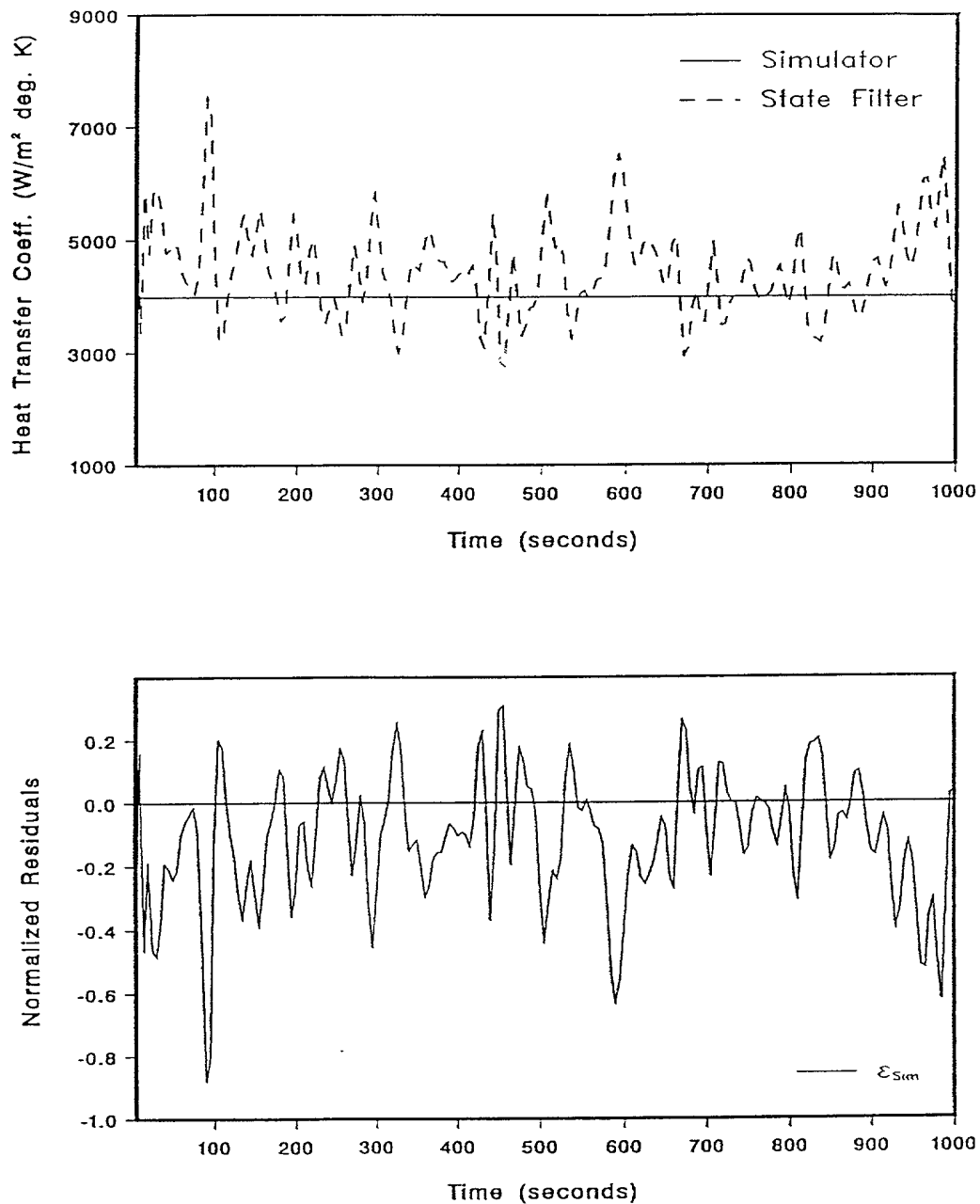


Figure 142. UTSG Process HTc Values, from the Simulator and NN State Filter, for the Adaptive NN HTc Filter 4 Using Test 6 (Step Input and High Noise Environment); (Top: U_{over} , Bottom: Normalized Residuals.)

It has nine states, three outputs, one controlled input and five input disturbances. The three outputs are: the downcomer water level, $L_w(t)$, the secondary saturated pressure, $P_s(t)$, and the cold-leg temperature, $T_{cl}(t)$. The controlled input is the feedwater flow rate, $W_{fw}(t)$. The five disturbances are: the hot-leg temperature, $T_{hl}(t)$, the primary fluid flow rate, $W_{pr}(t)$, the steam flow rate, $W_{st}(t)$, the feedwater temperature, $T_{fw}(t)$, and the primary fluid pressure, $P_{pr}(t)$. The UTSG system is open-loop unstable and so, for the transients conducted in this study, a controller is used to stabilize the system.

Two UTSG process models are used in this study: the UTSG scheduled model and the UTSG simulator itself (without process and measurement noise). The latter model is referred to as the UTSG simulator model. The scheduled model is obtained by linearizing the UTSG simulator at different power levels and then fitting the elements of the matrices describing the linear models with a variable that is a function of the operating power level of the UTSG system. Although the UTSG simulator model (without process and measurement noise) is a much more accurate model than the scheduled model, it is correspondingly more complex and computationally intensive and therefore less suitable for on-line use. Nevertheless, estimation schemes are developed with both these models to illustrate the importance of accurate models for estimation purposes.

Three state filters, Filters 1, 2, and 3, are developed to filter the riser void fraction, $\alpha_R(t)$, a state of the UTSG process system. Filter 1 and 2 are hybrid adaptive state filters and are based on the UTSG scheduled model and simulator model, respectively. Filter 3 is an adaptive state filter and is completely based on NNs. Filter 4 is also an adaptive state filter. However, Filter 4 is developed to estimate a constant parameter of the UTSG system, the primary heat transfer coefficient (HTC), U_{over} .

The performance of the developed state filters in Filters 1 to 4 is evaluated using validation data sets. The results of these tests are depicted in Tables 10, 11, 12 and 13. In Table 13 only the average $\epsilon_{Sim}(t)$ values are reported since the simulator model was used to develop Filter 4. Therefore, $\epsilon_{Mod}(t)$ is very close to $\epsilon_{Sim}(t)$. It is seen that

Table 10. UTSG Process System Hybrid Adaptive NN Riser Void Fraction Filter 1 Validation Test Results.

NN State Filter 1			NMSE	$\overline{\epsilon_{\text{Sim}}(t)}$	$\overline{\epsilon_{\text{Mod}}(t)}$
Hybrid Adaptive	Low Noise ($\sigma = 0.05$)	Ramp Inp.	3.2%	17.5%	1.6%
		Step Inp.	1.9%	14.2%	1.1%
	High Noise ($\sigma = 0.15$)	Ramp Inp.	4.4%	-12.1%	-1.2%
		Step Inp.	3.5%	12.1%	1.3%

Table 11. UTSG Process System Hybrid Adaptive NN Riser Void Fraction Filter 2 Validation Test Results.

NN State Filter 2			NMSE	$\overline{\epsilon_{\text{Sim}}(t)}$	$\overline{\epsilon_{\text{Mod}}(t)}$
Hybrid Adaptive	Low Noise ($\sigma = 0.05$)	Ramp Inp.	0.25%	4%	1%
		Step Inp.	0.05%	2.1%	0.8%
	High Noise ($\sigma = 0.15$)	Ramp Inp.	1.1%	8.3%	1.6%
		Step Inp.	0.2%	3.6%	0.9%

the performance of the state filter is good. However, the quality of the UTSG process model used to develop the state filters directly affects the quality (accuracy) of the resulting state filter values. For example, the accuracy of the state filter values in Filters 2 and 3 is much higher than those in Filter 1. This is a direct consequence of the fact that the scheduled process model is less accurate than the simulator model. The results in Filter 4 affirm that the state filtering methods developed can be applied to the problem of filtering constant or slowly varying parameters of the UTSG process.

Table 12. UTSG Process System Adaptive NN Riser Void Fraction Filter 3 Validation Test Results.

NN State Filter 3			NMSE	$\overline{\epsilon_{\text{Sim}}(t)}$	$\overline{\epsilon_{\text{Mod}}(t)}$
Adaptive	Low Noise ($\sigma = 0.05$)	Ramp Inp.	0.56%	0.38%	0.34%
		Step Inp.	0.05%	0.18%	0.12%
	High Noise ($\sigma = 0.15$)	Ramp Inp.	4.5%	-2.6%	-2.48%
		Step Inp.	0.11%	0.26%	0.24%

Table 13. UTSG Process System Adaptive NN HTC Filter 4 Validation Test Results.

NN State Filter 4			NMSE	$\overline{\epsilon_{\text{Sim}}(t)}$
Adaptive	Low Noise ($\sigma = 0.05$)	Test 1	1.2%	4%
		Test 2	0.8%	2.5%
		Test 3	0.5%	2.7%
		Test 4	0.55%	2.8%
		Test 5	0.46%	1.3%
		Test 6	0.58%	0.7%
		Test 7	0.3%	-0.85%
		Test 8	2.9%	-0.32%
	High Noise ($\sigma = 0.15$)	Test 5	0.6%	1.11%
		Test 6	1.28%	-2.9%

CHAPTER VIII

SUMMARY AND CONCLUSIONS

VIII.1 Summary of the Research Study

The objective of this research was to develop and demonstrate the feasibility of a method of nonadaptive and adaptive neural network (NN) state filtering for complex process systems, suitable for on-line applications. The developed NN state filtering methods was applied to three simulated systems: a Two-Input Two-Output (2I2O) system, a Motor-Pump process system, and a U-Tube Steam Generator (UTSG) process system. The simulated results obtained from these case studies were very encouraging, demonstrating the feasibility of the proposed method, and warranting further research.

In general, there are three types of state estimation problems: *prediction*, where an estimate of the state is made at a future time instant based on the system measurements at the present time instant; *filtering*, where an estimate of the state is made at the current time instant based on the system measurements made at the current time instant; and *smoothing*, where the estimate of the state is made at a past time instant based on the process measurements at the current time instant. State filtering methods in general, and adaptive state filtering, in particular, are very important in a number of industrial applications, such as inferential control, adaptive control, condition monitoring, fault diagnosis, data reconciliation etc. .

Modern estimation methods are dependent on mathematical models of the system studied. Therefore a model-based estimation method will be only as accurate as the system model on which it depends. As a result, the importance of obtaining high-fidelity system models for developing accurate estimation methods, cannot be over-stated. There are, in general, two types of mathematical models: *physical*, which

are derived from the laws of physics and/or chemistry governing the system, also referred to as first-principle models, and *empirical*, which are inferred by using system measurements only. Conventional estimation methods are largely based on linear system models. However, when dealing with complex systems, in general, and process systems, in particular, these linear models have a restricted range of validity. On the other hand, nonlinear models that are presently used, while being better than linear models, suffer from serious computational disadvantages when considered for on-line use. Recently, NNs have been shown to be effective at modeling complex process systems. In particular, NNs such as the Feedforward Multilayer Perceptron (FMLP) and the Recurrent Multilayer Perceptron (RMLP), have been shown to be accurate in single-step-ahead prediction (SSP) and multi-step-ahead prediction (MSP). Therefore, in this research, certain classes of NNs are used as nonlinear model structures in state estimation methods for complex process systems.

In Chapter II, a general overview of estimation methods based on linear models was given. Linear estimation methods as applied to static and dynamic systems were discussed. The most common method for solving estimation problems, the linear least-squares technique, was presented. The least-squares method is based on minimizing the square of an error term in order to determine the estimate(s). The recursive least-squares method is another way of applying the least-squares method, where the estimate is obtained as the system measurements are made. This is different than the batch least-squares method, where the estimate is obtained after a certain number of measurements have been collected. The recursive least-squares method is therefore more suitable for on-line applications, requiring updates of the estimates as new information becomes available.

The dynamic models described in Chapter II were divided to two types: input-output models and state-space models. Input-Output models are used to estimate the output of the system using a combination of current and past inputs and/or past outputs (measurements). The most common types of input-output models are the Auto-Regressive with eXogeneous inputs (ARX) and the Auto-Regressive Moving-Average with eXogeneous inputs (ARMAX) models. The SSP and MSP forms of

these two input-output models were derived. It was demonstrated that the MSP forms of these models can be recursively obtained from the SSP forms of the models by replacing the output measurements with the past predictions of the model itself.

State-space models represent the relationships between the input and output signals of a system, as a set of first-order differential or difference equations using an auxiliary *state* vector. The Kalman Filter (KF) is a method for recursively obtaining a state estimate of a system described by a linear state-space model. The KF algorithm can be split into two sequential steps: the *prediction step*, prior to the $(t + 1)$ -th measurement, where the SSP of the state and the output is obtained using the linear model equations, and the *update step*, following the $(t + 1)$ -th measurement, where the filtered value of the state is obtained by correcting the SSP of the state with the information obtained from the new output measurement. This correction is effected through the *innovations* term which is the difference between the current output measurement and the SSP of the output at the previous time instant. In view of the preceding discussion, the KF is characterized by the *prediction-update* equations reflecting the two-stage procedure for obtaining the filtered state values.

In general System Identification (SI) methods, both input-output and state-space, attempt to solve for the unknown system parameters using some form of a least-squares algorithm. These methods were discussed in Chapter II as applied to the ARX, the ARMAX and state-space models.

Adaptive estimation refers to estimation done when the model itself is unknown. Frequently, because of system drifts or different operating conditions, the available system model may prove to be inaccurate and must be updated. Therefore, a new model must be identified with the most current system measurements. Thus, in adaptive estimation methods, a model must be determined (identified) first, following which the estimation methods can be applied.

In Chapter III, a general overview of prediction methods based on nonlinear models was presented. As in Chapter II, both static and dynamic models were considered. In general, suitable nonlinear predictors are far more difficult to obtain than linear predictors. This is attributed to the intrinsic complexities of nonlinear models

and the paucity of appropriate analytical tools. In contrast, there is a wealth of such analytical tools for linear models. As in the case of linear models, the models used to obtain nonlinear predictors can be classified into two types: input-output models and state-space models. The most common types of nonlinear input-output models are the Nonlinear Auto-Regressive with eXogeneous inputs (NARX) and the Nonlinear Auto-Regressive Moving Average with eXogeneous inputs (NARMAX), which are analogous to the ARX and the ARMAX models. However, unlike the linear ARX and the linear ARMAX models, the relationships between the inputs and outputs of the NARX and the NARMAX models are nonlinear in nature. The SSP and MSP versions of the NARX and NARMAX predictors are also presented, followed by the general form of nonlinear state-space models.

The solution to most nonlinear estimation problems is based on a least-squares type algorithm. The NARX and the NARMAX models can be expanded in a polynomial-type functional form relating the inputs and the outputs. These models are referred to as the polynomial NARX and the polynomial NARMAX models. The polynomial expansion, representing the polynomial NARX and polynomial NARMAX models, is linear in the parameters. Therefore, a linear least-squares type estimation algorithm can be used to identify the parameters involved.

NNs, such as the Feedforward Multilayer Perceptron (FMLP) and the Recurrent Multilayer Perceptron (RMLP), are nonlinear model structures that have been shown effective in modeling complex process systems. Further, in addition to being nonlinear, the aforementioned NNs are nonparametric model structures. In fact, the FMLP and RMLP NNs can be shown to belong to the class of NARX models. NNs have also been shown to exhibit superior performance compared to polynomial NARX type models, especially when MSP performance is an evaluation criterion. Finally, fairly systematic SI methods exist that can identify the parameters of NN models. In NN terminology, SI is referred to as *training* and the NN model parameters are referred to as the *weights* and *biases* of the NN model.

In view of the stated potential of NNs in nonlinear estimation problems, the objective was then to investigate in some detail certain classes of NNs. Namely, the

well-known Feed-forward Multilayer Perceptron (FMLP) NN and the more recent Recurrent Multilayer Perceptron (RMLP) NN were discussed. The FMLP comprises successive layers of nodes, where the nodes of a particular layer are connected to the nodes of the next layer by weighted links. This is to say that signals passing from one node to another are multiplied by a weighting factor, the weight. Then, the resulting signals are summed together, and finally they are transformed by a nonlinear function, the squashing function. The output signal of this nonlinear function is the output of that particular node. This process is repeated for all the nodes.

The RMLP is similar to the FMLP, but it is also different in some important aspects. The RMLP has a feedforward part, and like the FMLP, that performs nonlinear curve-fitting. In addition, however, unlike the FMLP, the RMLP has a one-time-delayed cross-talk and local recurrency connections in the hidden layers, which serve as local memory capturing temporal correlations. These recurrencies in the RMLP account for its superior modeling properties when dealing with certain types of complex systems.

Training of the NN, either the FMLP or RMLP, refers to the process of determining its weights and biases. The *estimation data set* is the data collected from the actual system or the simulator that is used in the training. Further, the estimation data set is divided into two: the part used in weight update and the part used in cross-validation. The former is used to actually train the NN, whereas the latter is used in determining when the training is sufficient and must be terminated. The *validation data set* is the data used to evaluate the performance of the NN after the training is completed. This data set should be a true indication of the generalization properties of the NN.

The most widely used method for training an FMLP is the Backpropagation (BP) algorithm and its variants. The BP algorithm is basically a least-squares algorithm that attempts to minimize an error criterion by adjusting the weights of the network. One of the training algorithms developed for training the RMLP network is the dynamic gradient descent learning algorithm, which, as its name implies, is a gradient descent method. The basic mechanism behind this learning rule is the

adjustment of the network weights and the bias terms, until the Normalized-Mean-Squared Error (NMSE) between the output predicted by the network and the target output satisfies a certain stopping criterion.

Two learning algorithms for the RMLP were considered, based on dynamic gradient descent, the Teacher Forcing (TF) algorithm and the Global Feedback (GF) algorithm. TF means that the past outputs present in the network input layer consist of the latest sensed output(s) of the system. In other words, at each time step the network input layer is updated using the latest sensed input(s) and output(s). GF on the other hand, means that the past outputs present in the network input layer consists of past output(s) predicted by the NN itself. The fact that predicted output(s) are utilized by the network in GF learning, as compared to sensed output(s) in TF learning, makes the derivation of the gradients in the former more complex than in the latter. Moreover, in GF learning, the network learns the input-output data based on its own MSP only. This feature enables the network to perform well in MSP during training. By designing such a MSP predictor, good performance is also obtained on other data sets not seen during the training.

Once the details of the FMLP and the RMLP NN architectures, their associated learning algorithms, and their use as nonlinear predictors were discussed in Chapter IV, the framework for NN state filtering methods was developed. The conventional method for nonlinear state filtering is referred to as the Extended Kalman Filter (EKF). In this approach, the nonlinear system model is linearized about the estimated state values and the standard KF algorithm is applied. This method, while having been successfully applied to a number of problems, has been known to exhibit convergence problems.

The general approach adopted in developing the NN state filtering method was to follow a KF type approach. In particular, the two-stage approach of formulating and solving the prediction-update equations, as is done in the standard KF and in the EKF, is adopted. However, the novelty of the proposed NN state filtering method is that the prediction-update equations will be, in general, nonlinear. The positive aspect of this extension is in the ability to apply the method to arbitrarily complex

filtering problems. The negative aspect, however, is the lack of any guarantee for convergence.

Nonadaptive and adaptive NN state filtering methods were developed. In the nonadaptive methods, it is assumed that a system model is available. The system model takes on the role of the predictor and a NN, either an FMLP or an RMLP, provides the update function. This NN is referred to as the NN state filter. In this way, the KF type prediction-update equations are specified. Importantly, these equations are nonlinear in nature and no linearization is involved as in the EKF. In the adaptive state filtering method, it is assumed that a system model is unavailable. Therefore, a predictor is identified. This is accomplished by using two NNs as the predictor: one NN is used for the output prediction and the other NN is used for the state prediction. Therefore, the system model (predictor) in the nonadaptive state filtering method is replaced by two NN predictors in the adaptive filtering method. The update function is still provided by a NN, the NN state filter, as in the nonadaptive method. In this way, the adaptive state filtering method is entirely empirical, because it consists of only NNs specifying the prediction-update equations. In case a system model is available, but it is considered inaccurate because of deterministic modeling uncertainties, a NN “error model” is used to correct for these uncertainties. In this research, the method of using the available system model and the NN “error model”, together with the NN state filter, is referred to as a hybrid adaptive state filter.

In the remainder of the dissertation, the nonadaptive, adaptive, and hybrid NN state filtering methods developed in Chapter IV are applied to three simulated systems of increasing complexity: an artificial 2I2O system, a DC Motor-Pump process system, and a UTSG process system. All the developments and testing of the NN state filters was performed on simulators, in lieu of actual process measurements.

In Chapter V, the NN nonadaptive and adaptive state filtering method was applied to the 2I2O system. This is a simple nonlinear system and it comprises three states, two inputs and two outputs. In this application, the objective of the state filtering was to filter one of the three states of the 2I2O system. Validation tests of

the developed state filter indicate that the proposed state filtering method performs quite well.

In Chapter VI, the NN hybrid adaptive and adaptive state filtering method were applied to the DC Motor-Pump system. The DC Motor-Pump system is a relatively complex system, but simple enough to be modeled fairly accurately. In this application, the objective of the state filtering was to filter the armature resistance, which is modeled as a state of the DC Motor-Pump system. In addition, the hybrid adaptive state filtering method was used to filter both the armature resistance and the magnetic flux linkage, a constant parameter. Extensive validation tests of the developed state filters demonstrate that, again, the proposed state filtering methods performed quite well for a wide range of validation tests.

In Chapter VII, the NN hybrid adaptive and adaptive state filtering method was applied to the UTSG process system. The UTSG process is a complex process system with one control input, five disturbances, and three measured outputs. An existing UTSG process simulator is used to generate data to develop and test the state filters. In addition, two different UTSG process models are used to develop the state filters: the UTSG scheduled model and the simulator model. The UTSG scheduled model is based on several linear UTSG models, whose parameters are changed depending on one parameter (the scheduling variable) that reflects the changing UTSG operating conditions. The UTSG simulator model is the UTSG simulator itself without the process and measurement noise. The UTSG simulator model is a more accurate representation of the UTSG process system than the scheduled model.

In this application, the objective of the state filtering was to filter the riser void fraction. The riser void fraction is a critical process variable that cannot be measured. Hybrid adaptive NN state filters are developed using both the UTSG scheduled model and the simulator model. In the development of the adaptive state filters, the state information was obtained from the UTSG simulator model.

In addition to filtering the riser void fraction, an adaptive state filter was developed for the primary heat transfer coefficient (HTC) of the UTSG. The HTC is another unmeasurable critical process parameter that is slowly varying. However,

for practical purposes, the HTC can be considered as constant for most UTSG operational transients. All of the developed state filters were evaluated with test data independent of the estimation data. The results of the evaluation tests indicate that the accuracy of the state filters is strongly dependent on the process model used to develop the state filter. Therefore, as expected, the state filters based on the UTSG scheduled model will be less accurate as compared with the state filters based on the UTSG simulator model.

In summary, the application results from these three increasingly more complex process systems indicate that the proposed state filtering method is accurate enough to be considered a serious candidate for difficult industrial estimation problems. The case studies indicate that the proposed algorithms for implementing the state filters are computationally feasible, and strongly dependent upon the system model used in their development. Therefore, considering the complexity of the process systems under consideration, the adaptive form of the proposed state filtering method is the most likely approach to make a potentially significant contribution towards the solution of real-world estimation problems.

VIII.2 Conclusions from the Research Study

The objectives of this research has been to develop and demonstrate the feasibility of a method for designing accurate adaptive state filters for complex process systems using NNs, and in particular, recurrent networks. This objective was achieved by using a KF type approach, combined with nonlinear and nonparametric system models. These models consisted of FMLP and RMLP NNs. Further, the NNs were developed with GF, at the training and validation stages and specific guidelines were followed to reduce network complexity to prevent overfitting. The design of the NNs in the state filters with MSP performance as an objective, resulted in enhanced state estimates.

The conclusions drawn from this research can be summarized as follows:

- (1) High-fidelity first-principles system models are required in order to develop high-fidelity state filters. State information from the high-fidelity system models is critical in developing both the nonadaptive and adaptive NN state filters. Therefore, the fidelity of the state estimates will only be as good the fidelity of the state values obtained from the system model.
- (2) In case a high-fidelity system model is not available, then state information must be obtained experimentally using a test-bed using the system under study. In short, accurate state information, either from a first-principles system model or from experiments, is required in order to develop high fidelity state filters.
- (3) If first-principle system models cannot be used as on-line predictors, or if they are not available, then empirical NN predictors appear to be the best available alternative.
- (4) High fidelity NN predictors for use in adaptive NN state filters can be obtained by using GF during training, and by selecting MSP as a performance criterion in neural network design.
- (5) The critical step in designing accurate adaptive state filters using NNs is to follow the fundamental principles of Kalman Filtering, regarding the two separate steps in filtering, i.e. the predictor and then the update, and the information flow between these two steps.

VIII.3 Recommendations for Future Research Directions

The NN state filtering method developed in this research is only a starting point for further investigations into adaptive state filtering methods for complex systems, and their applications to industrial problems. Based on the research reported in this dissertation, some possible topics for future research are:

- (1) Applications of nonlinear state filtering methods to inferential control, condition monitoring and fault diagnosis.
- (2) Investigation of methods for estimating the error covariance matrix in state filtering problems for use in determining the on-line performance of the designed state filters.

- (3) Investigation of on-line adaptation methods for state filtering.
- (4) Investigation of state filtering methods for system states that are “weakly” observable.
- (5) Further refinement of learning algorithms for NNs, both the FMLP and RMLP, with GF.

REFERENCES

- [1] B. D. O. Anderson, and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.
- [2] K. J. Åström, "Maximum Likelihood and Prediction Error Methods," *Automatica*, vol. 16, pp. 551-574, 1980.
- [3] A. F. Atiya, Personal Communication, Cairo University, Egypt, 1993.
- [4] A. R. Barron, "Universal Approximation Bounds for Superpositions of a Sigmoidal Function," *IEEE Transactions on Information Theory*, vol. 39, pp. 930-945, 1993.
- [5] A. R. Barron, "Approximation and Estimation Bounds for Artificial Neural Networks," *Journal of Machine Learning*, vol. 14, pp. 115-133, 1994.
- [6] M. Basseville, "Detecting Changes in Signals and Systems - A Survey," *Automatica*, vol. 26, pp. 309-326, 1988.
- [7] S. A. Billings, "Identification of Nonlinear Systems - A Survey," *IEE Proc. Control Theory and Applications*, vol. 5, pp. 272-285, 1980.
- [8] S. A. Billings, S. Chen, and A. Korenberg, "Identification of MIMO Nonlinear Systems Using a Forward-regression Orthogonal Estimator," *International Journal of Control*, vol. 49, pp. 2157-2189, 1989.
- [9] S. A. Billings, H. B. Jamaluddin, and S. Chen, "Properties of Neural Networks with Applications to Modelling Nonlinear Dynamical Systems," *International Journal of Control*, vol. 55, pp. 193-224, 1992.
- [10] S. Chen, and S. A. Billings, "Modeling and Analysis of Nonlinear Time Series," *International Journal of Control*, vol. 50, pp. 2151-2171, 1989.
- [11] S. Chen, and S. A. Billings, "Representations of Nonlinear Systems: the NARMAX Model," *International Journal of Control*, vol. 49, pp. 1013-1032, 1989.
- [12] S. Chen, and S. A. Billings, "Recursive Prediction Error Parameter Estimator for Nonlinear Models," *International Journal of Control*, vol. 49, pp. 569-594, 1989.
- [13] S. Chen, and S. A. Billings, "Orthogonal Least Squares Methods and Their Application to Nonlinear System Identification," *International Journal of Control*, vol. 50, pp. 1873-1896, 1989.

- [14] S. Chen, S. A. Billings, and P. M. Grant, "Nonlinear System Identification Using Neural Networks," *International Journal of Control*, vol. 51, pp. 1191-1214, 1990.
- [15] J. I. Choi, "Nonlinear Digital Computer Control for the Steam Generator System in a Pressurized Water Reactor Plant," Ph. D. dissertation, MIT, Cambridge, MA, August 1987.
- [16] K. T. Chong, "Nonlinear Dynamic System Identification Using Neural Networks," Ph. D. dissertation, Texas A&M University, College Station, TX, August, 1992.
- [17] R. N. Clark, "Instrument Fault Detection in a Pressurized Water Reactor Pressurizer," *Nuclear Technology*, vol. 56, pp. 23-32, 1982.
- [18] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, vol. II, pp. 133-141, June 1990.
- [19] J. P. DeCruyenaere, and H. M. Hafez, "A Comparison Between Kalman Filters and Recurrent Neural Networks," *Proceedings of IJCNN-92*, Baltimore, MD, vol. IV, pp. 247-251, 1992.
- [20] S. J. Eckert, K. A. Loparo, and Z. S. Roth, "An Application of Nonlinear Filtering to Instrument Failure Detection in a Pressurized Water Reactor," *Nuclear Technology*, vol. 74, pp. 139-151, 1986.
- [21] B. Fernandez, A. G. Parlos, and W. K. Tsai, "Nonlinear System Identification using Artificial Neural Networks," *Proceedings of the 1990 International Joint Conference on Neural Networks*, vol. II, pp. 133-141, June 1990.
- [22] J. W. Forrester, *Urban Dynamics*, MIT Press, Cambridge, MA, 1969.
- [23] P. M. Frank, "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy - A Survey and Some New Results," *Automatica*, vol. 26, pp. 459-474, 1990.
- [24] C. E. Garcia, D. M. Prett, and M. Morari, "Model Predictive Control: Theory and Practice - A Survey," *Automatica*, vol. 25, pp. 335-348, 1989.
- [25] K. F. Gauss, *Theory of the Motion of the Heavenly Bodies Moving About the Sun in Conic Sections*, translated from *Theoria Motus Corporum Coelestium*, 1809, by Charles Henry Davis, 1847, Dover, New York, NY, 1857.

- [26] G. Geiger, "Monitoring of an Electrical Driven Pump Using Continuous-Time Parameter Estimation Methods," *Identification and System Parameter Estimation - Proceedings of the Sixth IFAC Symposium*, vol. 1, pp. 603-608, 1982.
- [27] G. Geiger, "Fault Identification of a Motor-Pump System using Parameter Estimation and Pattern Classification," *IFAC 9th Triennial World Congress*, Budapest, Hungary, 1984.
- [28] A. Gelb, *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.
- [29] G. C. Goodwin, and K. S. Sin, *Adaptive Filtering Prediction and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- [30] M. J. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing Company, Boston, MA, 1996.
- [31] R. Harber, and H. Unbehauen, "Structure Identification of Nonlinear Dynamic Systems - A Survey on Input/Output Approaches," *Automatica*, vol. 26, No 4-A, pp. 651-677, 1990.
- [32] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- [33] S. Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press, Piscataway, NJ, 1994.
- [34] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Science U.S.*, vol. 79, pp. 2554-2558, April 1982.
- [35] R. Isermann, "Process Fault Detection Based on Modeling and Estimation Methods - A Survey," *Automatica*, vol. 20, pp. 387-404, 1984.
- [36] R. Isermann, "Process Fault Diagnosis with Parameter Estimation Methods," *IFAC Digital Computer Applications to Process Control*, pp. 51-60, Vienna, Austria, 1985.
- [37] R. Isermann, "Process Fault Diagnosis Based on Process Model Knowledge-Part 1: Principles for Fault Diagnosis with Parameter Estimation," *Transactions of the ASME: Journal of Dynamic Systems, Measurement and Control*, vol. 113, pp. 620-626, 1991.
- [38] J.-N. Juang, *Applied System Identification*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

- [39] R. E. Kalman, and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," *Trans. ASME D.J. Basic Eng.*, vol. 83, pp. 95-107, 1961.
- [40] M. Kitamura, and E. Türkcan, "Empirical Modeling Approach To Fault Detection and Identification in Nuclear Power Plant," *IFAC Identification and System Parameter Estimation*, pp. 693-698, 1985.
- [41] D. G. Lainiotis, and K. N. Plataniotis, "Neural Network Estimators: Application To Ship Position Estimation," *Proceedings of the 1994 IEEE International Conference on Neural Networks*, Orlando, FL, pp. 4710-4717, 1994.
- [42] D. G. Lainiotis, and K. N. Plataniotis, "Adaptive Dynamic Neural Network Estimators," *Proceedings of the 1994 IEEE International Conference on Neural Networks*, Orlando, FL, pp. 4736-4745, 1994.
- [43] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, Hertfordshire, UK, 1987.
- [44] L. Ljung, and T. Glad, *Modeling of Dynamic Systems*, Prentice-Hall, Hertfordshire, UK, 1994.
- [45] L. Ljung, and T. Sjöberg, "A System Identification Perspective on Neural Nets," *IEEE Workshop on Neural Networks for Signal Processing*, May 1992.
- [46] J. T. Lo, "Optimal Filtering by Recurrent Neural Networks," United States Patent, Patent No. 5,408,424, April 1995.
- [47] T. E. Marlin, *Process Control - Designing Processes and Control Systems for Dynamic Performance*, McGraw-Hill, New York, NY, 1995.
- [48] S. K. Menon, and A. G. Parlos, "Gain-Scheduled Nonlinear Control of U-Tube Steam Generator Water Level," *Nuclear Science and Engineering*, vol. III, no. 3, pp. 294-308, 1992.
- [49] K. S. Narendra, and K. Parthasarathy, "Identification and Control of Dynamic System Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 4-27, 1990.
- [50] K. S. Narendra, and K. Parthasarathy, "Neural Networks in Control Systems," Workshop on Neural Networks in Control Systems, *American Control Conference*, Workshop Manual, 1991.
- [51] K. S. Narendra, and W. L. Parthasarathy, "Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252-262, 1991.

- [52] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, New York, NY, 1990.
- [53] G. F. Page, J. B. Gomm, and D. Williams, (eds.), *Application of Neural Networks to Modelling and Control*, Chapman & Hall, London, 1993.
- [54] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, NY, 1991.
- [55] A. G. Parlos, K. T. Chong, and A. F. Atiya, "U-Tube Steam Generator Empirical Model Development and Validation Using Neural Networks," *ANS Transactions*, vol. 65, pp. 108-109, June 1992, Boston, MA.
- [56] A. G. Parlos, A. F. Atiya, K. T. Chong, B. Fernandez, and W. K. Tsai, "Nonlinear Identification of Process Dynamics Using Neural Networks," *Nuclear Technology*, vol. 97, pp. 79-95, 1992.
- [57] A. G. Parlos, K. T. Chong, and A. F. Atiya, "Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics," *IEEE Transactions on Neural Networks*, vol. 5, pp. 255-266, 1994.
- [58] A. G. Parlos, K. T. Chong, and A. F. Atiya, "U-Tube Steam Generator Empirical Model Development and Validation Using Neural Networks," *ANS Transactions*, Boston, MA, vol. 65, pp. 108-109, June 1992.
- [59] R. Patton, P. M. Frank, and R. N. Clark (eds.), *Fault Diagnosis in Dynamic Systems - Theory and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [60] W. Pitts, and W. S. McCulloch, "How We Know Universals: The Perception of Auditory and Visual Forms," *Bulletin of Mathematical Biophysics*, vol. 9, pp. 127-147, 1947.
- [61] F. J. Pineda, "Generalization of Backpropagation to Recurrent Neural Networks," *Physical Review Letters*, vol. 59, pp. 2229-2232, 1987.
- [62] F. J. Pineda, "Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation," *Neural Computation*, vol. 1, pp. 161-172, 1989.
- [63] N. F. Portmann, D. Lindhoff, G. Sorgel, and O. Gramckow, "Application of Neural Networks in Rolling Mill Automation," *Iron and Steel Engineer*, pp. 33-36, February 1995.
- [64] G. V. Puskorius, and L. A. Feldkamp, "Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 279-297, 1994.

- [65] O. T. Rais, "Modeling Complex Process Systems Using The Recurrent Multi-layer Perceptron," Ph. D. dissertation, Texas A&M University, College Station, TX, August 1995.
- [66] I. B. Rhodes, "A Tutorial Introduction to Estimation and Filtering," *IEEE Transactions on Automatic Control*, vol. AC-16, no. 6, pp. 668-706, 1971.
- [67] F. Rosenblatt, "The Perceptron; A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Reviews*, vol. 65, pp. 386-408, 1958.
- [68] D. E. Rumelhart, and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, I, & II*, MIT Press, Cambridge, MA, 1986.
- [69] F. C. Schweppe, *Uncertain Dynamic Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
- [70] J.-J. E. Slotine, *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- [71] T. Söderström, and P. Stoica, *System Identification*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [72] H. W. Sorenson, "Least-Squares Estimation: From Gauss to Kalman," *IEEE Spectrum*, pp. 63-68, July 1970.
- [73] V. Strejc, "Least Squares Parameter Estimation," *Automatica*, vol. 16, pp. 535-550, 1980.
- [74] W. H. Strohmayer, "Dynamic Modeling of Vertical U-Tube Steam Generators for Operational Safety Systems," Ph. D. dissertation, MIT, Cambridge, MA, August 1982.
- [75] H.-T. Su, T. J. McAVoy, and P. Werbos, "Long Term Predictions of Chemical Processes Using Recurrent Neural Networks : A Parallel Training Approach," *Ind. Eng. Chem. Res.*, vol. 31, pp. 1338-1352, 1992.
- [76] P. Swerling, "Modern State Estimation Methods from the Viewpoint of the Method of Least Squares," *IEEE Transactions on Automatic Control*, vol. AC-16, no. 6, pp. 707-719, 1971.
- [77] H. Tanizaki, *Nonlinear Filters - Estimation and Applications*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, 1993.

- [78] N. E. Todreas, and M. S. Kazimi, *Nuclear Systems I*, Hemisphere Publishing Corporation, New York, NY, 1990.
- [79] L. S. Tong, *Principles of Design Improvement for Light Water Reactors*, Hemisphere, New York, NY, 1988.
- [80] A. C. Tsoi, and A. D. Back, "Locally Recurrent Globally Feedforward Networks: A Critical Review of Architectures," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 229-239, 1994.
- [81] J. L. Tylee, "Estimation of Failed Sensor Outputs," *Nuclear Science and Engineering*, vol. 96, pp. 145-152, 1987.
- [82] J. L. Tylee, "On-Line Failure Detection in Nuclear Power Plant Instrumentation," *IEEE Transactions on Automatic Control*, vol. 28, no. 3, March 1983.
- [83] K. Watanabe, and D. M. Himmelblau, "Fault Diagnosis in Nonlinear Chemical Processes- Part 1 and 2," *AIChE Journal*, vol. 29, no. 2, 1983.
- [84] K. Watanabe, and D. M. Himmelblau, "Incipient Fault Diagnosis of Nonlinear Processes with Multiple Causes of Faults," *Chemical Engineering Science*, vol. 39, no. 3, pp. 491-508, 1984.
- [85] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of the IEEE*, vol. 78, pp. 1550-1560, 1990.
- [86] P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph. D. dissertation, Harvard University, Cambridge, MA, 1974.
- [87] D. A. White, and D. A. Sofge, *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York, NY, 1992.
- [88] N. Wiener, *Cybernetics : or Control and Communication in the Animal and the Machine*, MIT Press, Cambridge, MA, 1948.
- [89] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Applications*, MIT Press, Cambridge, MA, 1949.
- [90] R. J. Williams, and D. Zipser, "A Learning Algorithm for Continually Running fully Recurrent Neural Networks," *Neural Computation*, vol. 1, pp. 270-280, 1989.

- [91] R. J. Williams, and J. Peng, "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories," *Neural Computation*, vol. 2, pp. 490-501, 1990.
- [92] R. J. Williams, A. Davies, and P. R. Drake, *Condition-based Maintenance and Machine Diagnostics*, Chapman and Hall, London, UK, 1994.
- [93] R. J. Williams, "Training Recurrent Networks Using the Extended Kalman Filter," *Proceedings of IJCNN-92*, Baltimore, MD, vol. IV, pp. 241-246, 1992.
- [94] A. S. Willsky, "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, vol. 12, pp. 601-611, 1976.
- [95] A. S. Willsky, E. Y. Chow, S. B. Gershwin, C. S. Greene, P. K. Houpt, and A. L. Kurkjian, "Dynamic Model-Based Techniques for the Detection of Incidents on Freeways," *IEEE Transactions on Automatic Control*, vol. AC-25, no. 3, 1980.

APPENDIX A

BACKPROPAGATION-THROUGH-TIME ALGORITHM

A.1 Introduction

Backpropagation (BP) and numerous of its variants are currently the most widely used algorithm for training perceptron-type NNs. The basic steps involved in BP learning are the evaluation of the error gradient term where the error is usually defined as the square of the difference between the NN output and the target value. The error gradients are computed with respect to each weight and bias of the NN. Then adjustment of the weights by a term that is proportional to the error gradients follows. These steps ensure that the error term is minimized, at least locally, and hence the NN output approaches the target values. An implicit assumption in the error gradient calculation is that the inputs to the NN are variables that are *independent* of the weights. However, when Global feedback (GF) is used, the delayed outputs of the NN are fed back and are supplied as inputs to the NN itself. Therefore, some of the inputs to the NN namely the past outputs of the NN itself, are dependent on the weights of the NN. Therefore, the “traditional” BP algorithm cannot be applied directly, and it must account for the dependencies of the NN inputs to the NN weights.

The Backpropagation-Through-Time (BTT) algorithm is such a method of accounting for the dependencies of the NN inputs (past outputs of the NN itself) to the NN weights. It is believed that by accounting for such dependencies, the “memory” of the NN will be enhanced, and thereby the overall modeling capabilities of the NN will be improved.

This appendix will outline some of the principles of the BTT algorithm. For further details, the reader is referred to Werbos [75], [85].

642

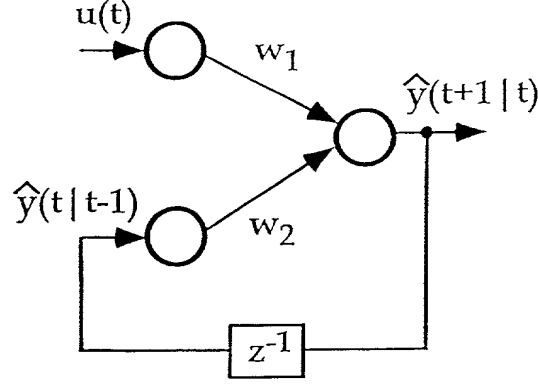


Figure 143. Simple Feedforward Neural Network with Global Feedback.

A.2 The BTT Algorithm

First, for simplicity, the BTT algorithm is explained using a two-layer, two input, single output, feedforward NN with GF. The two inputs to the network are $u(t)$ and $\hat{y}(t|t-1)$. The output of the network is $\hat{y}(t+1|t)$. The weights, w_1 and w_2 , link the two inputs to the hidden node. Assume that there is no bias term in the hidden node. Figure 143 shows this feedforward network. The targets for the training are the outputs $y(1), \dots, y(t)$, for $1 \leq t \leq N$, where N is the total number of training data points.

The error term, E , that is to be minimized is expressed as:

$$E = \frac{1}{2} \sum_{t=0}^N E(t+1) \equiv \frac{1}{2} \sum_{t=0}^N [\hat{y}(t+1|t) - y(t+1)]^2, \quad (299)$$

and the weights are updated using the following rule:

$$\Delta w_i = -\eta \sum_{t=0}^K \left(\frac{\partial E(t+1)}{\partial w_i} \right), \quad (300)$$

where $K = 1$ for individual update and $K = N$ for batch update. The output of the NN, $\hat{y}(t+1|t)$, can be expressed as:

$$\hat{y}(t+1|t) = \sigma(w_1 u(t) + w_2 \hat{y}(t|t-1)), \quad (301)$$

where $\sigma(\cdot)$ is the discriminatory function.

The gradient of the error term with respect to weight w_1 is obtained by differentiating equation (299). The following expression results:

$$\frac{\partial E(t+1)}{\partial w_1} = [\hat{y}(t+1|t) - y(t+1)] \frac{\partial \hat{y}(t+1|t)}{\partial w_1}. \quad (302)$$

The $\frac{\partial \hat{y}(t+1|t)}{\partial w_1}$ term in equation (302) is obtained by differentiating equation (301).

The resulting expression is as follows:

$$\frac{\partial \hat{y}(t+1|t)}{\partial w_1} = \sigma' \left(w_1 u(t) + w_2 \hat{y}(t|t-1) \right) \left[u(t) + w_2 \frac{\partial \hat{y}(t|t-1)}{\partial w_1} \right]. \quad (303)$$

From equation (303), it is seen that the expression for $\frac{\partial \hat{y}(t+1|t)}{\partial w_1}$ involves the gradient $\frac{\partial \hat{y}(t|t-1)}{\partial w_1}$ from the *previous* time step. The expression for the gradient $\frac{\partial \hat{y}(t|t-1)}{\partial w_1}$, using equation (303), is as follows:

$$\frac{\partial \hat{y}(t|t-1)}{\partial w_1} = \sigma' \left(w_1 u(t-1) + w_2 \hat{y}(t-1|t-2) \right) \left[u(t-1) + w_2 \frac{\partial \hat{y}(t-1|t-2)}{\partial w_1} \right]. \quad (304)$$

From equation (304), it is seen that the gradient $\frac{\partial \hat{y}(t|t-1)}{\partial w_1}$ involves the gradient at the time instant $(t-1)$, $\frac{\partial \hat{y}(t-1|t-2)}{\partial w_1}$. This pattern continues until the initial time instant, $t=0$, where $\frac{\partial \hat{y}(0|-1)}{\partial w_1}$ is set to zero.

From the preceding discussion, it is seen that the gradient calculations in equations (303), and consequently equation (302), require the values of the corresponding gradients at *every* preceding time instant until the initial time, $t=0$, is reached. In other words, the error gradient calculation is *backpropagated through time*. A similar derivation can be made for the gradient calculations with respect to w_2 , and this is not reported here.

From the preceding example of a simple feedforward NN, the basic principle behind the BTT algorithm is introduced: the error gradient calculations at a time instant $(t+1)$ are dependent on gradient calculations at every preceding time instant, $t, (t-1), \dots, 1$.

Now, consider a general three-layer feedforward NN with n_i inputs, $u(t)$, n_y outputs, $\hat{y}(t+1|t)$, and n_h nodes in the hidden (second) layer. Further, the feedforward

NN has GF and therefore the total number of inputs to the NN are $n_i + n_y$ (n_i “independent” inputs and n_y past outputs fed back to the network). The error term, E , to be minimized can be expressed as:

$$E = \frac{1}{2} \sum_{t=0}^N E(t+1) \equiv \frac{1}{2} \sum_{t=0}^N \sum_{j=1}^{n_y} [\hat{y}_j(t+1|t) - y_j(t+1)]^2, \quad (305)$$

and the weights in vectorial form, \mathbf{w} , are updated using the following rule:

$$\Delta \mathbf{w} = -\eta \sum_{t=0}^K \left(\frac{\partial E(t+1)}{\partial \mathbf{w}} \right), \quad (306)$$

where, $K = 1$ for individual update and $K = N$ for batch update.

Now, the error gradient to be calculated in equation (303) can be expressed as:

$$\frac{\partial E(t+1)}{\partial \mathbf{w}} = [\hat{\mathbf{y}}(t+1|t) - \mathbf{y}(t+1)] \frac{\partial \hat{\mathbf{y}}(t+1|t)}{\partial \mathbf{w}}. \quad (307)$$

The partial derivative $\frac{\partial \hat{\mathbf{y}}(t+1|t)}{\partial \mathbf{w}}$ in equation (307) can be expressed as follows:

$$\frac{\partial \hat{\mathbf{y}}(t+1|t)}{\partial \mathbf{w}} = \frac{\partial \hat{\mathbf{y}}(t+1|t)}{\partial \hat{\mathbf{y}}(t|t-1)} \frac{\partial \hat{\mathbf{y}}(t|t-1)}{\partial \mathbf{w}}. \quad (308)$$

The first partial derivative on the right-hand-side of equation (308) is the derivative of the NN output with respect to some of the *inputs* of the NN. To evaluate the second partial derivative in equation (308), it is convenient to consider the inputs to the NN, $\hat{\mathbf{y}}(t|t-1)$, as the outputs of another NN. The inputs to this network, $\hat{\mathbf{y}}(t-1|t-2)$, are the outputs of a third network, and so on. This is referred to as *unfolding* of the NN. A schematic diagram of a NN unfolded in time is shown in Figure 144. From Figure 144 it is seen that there are $(t+1)$ NNs; one network corresponding to each time instant from 0 to t . Each of these networks are duplicates of the NN at time t , with the same weights. Therefore, the gradient $\frac{\partial \hat{\mathbf{y}}(t|t-1)}{\partial \mathbf{w}}$ is calculated in the NN corresponding to the time instant $(t-1)$ using the equation (308) expressed for $\frac{\partial \hat{\mathbf{y}}(t|t-1)}{\partial \mathbf{w}}$ in terms of $\frac{\partial \hat{\mathbf{y}}(t-1|t-2)}{\partial \mathbf{w}}$. This is repeated till the last NN (corresponding to the time instant $t=0$) is reached. In this way, the error gradients are backpropagated through time.

645

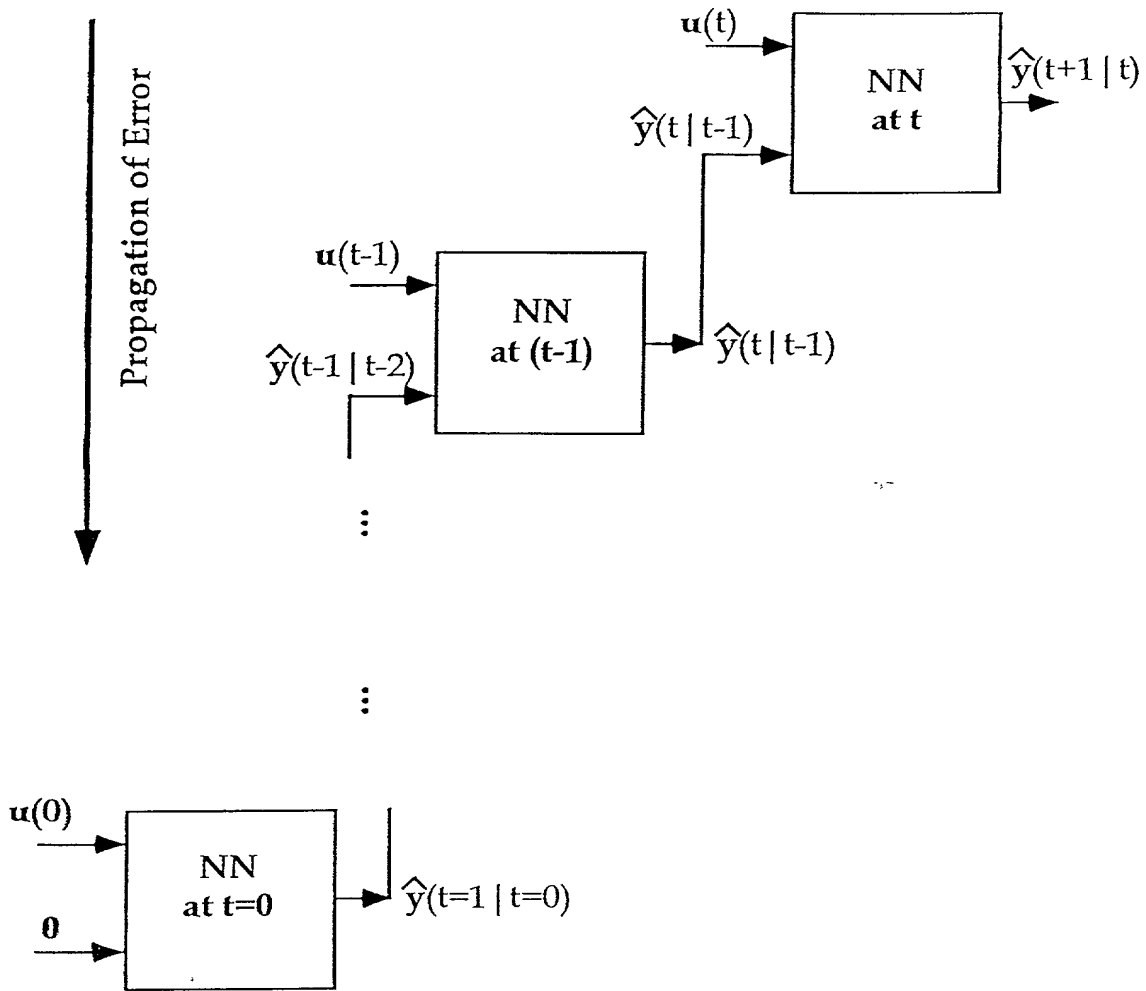


Figure 144. Unfolding of the Neural Network Across Time.

646
VITA

Sunil Kumar Menon was born on February 9, 1967 in Trichur, Kerala-India. He obtained a B.E (with honors) degree in Electrical and Electronics Engineering at the Regional Engineering College, Tiruchirapalli, India in 1988. Mr. Menon subsequently joined the Nuclear Engineering department at Texas A&M University where he obtained an M.S. degree in 1990. He continued his education at the same department and university and will obtain a Ph. D degree in August, 1996.

During the course of his education, Mr. Menon confused the dictum *sapere aude* with *Elvem regis* and made several trips to Graceland. A bad sky-dive over Las Vegas, however, brought Mr. Menon back to earth. In addition to collecting theories on alien abductions, Mr. Menon firmly believes that the world should be ruled by cats. His permanent address is: A-4, 33 Halls Road, Egmore, Madras-8, India.

646-33 Halls Road, Egmore, Madras-8, India

647

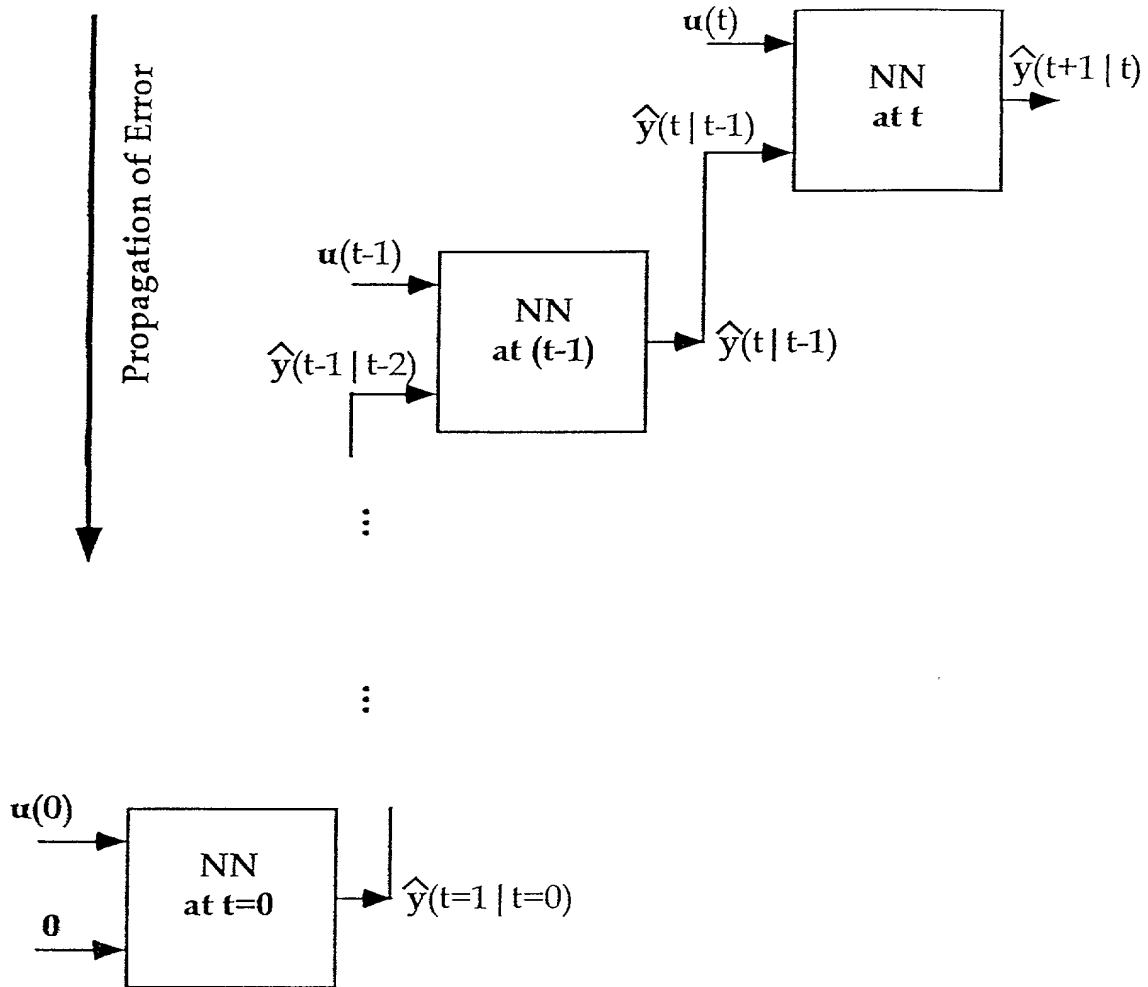


Figure 144. Unfolding of the Neural Network Across Time.

ABSTRACT

Long-Term Load Forecasting in Electric Power Industry. (August 1997)

Esmail Oufi, B.S.; M.S., Kuwait University

Co-Chairs of Advisory Committee: Dr. Alton D. Patton
Dr. Alexander G. Parlos

The objective of this research study is to develop methods to improve long-term load forecasting practice in electric power industry. This objective is accomplished through the development of more accurate long-term load forecasts as well as more accurate assessment of the uncertainties involved.

Scarcity of the historical observations imposes restrictions on effective solutions of the long-term load forecasting problem. This study shows that nonlinear estimators of controlled complexity are viable tools in long-term load forecasting. The estimator complexity level should be large enough to capture the general historical trends. However, it should be small enough to avoid memorization of all the historical events. A Monte Carlo filtering process is proposed to optimize different estimator structures and parameters, and to select a level of complexity. Forecasts developed using the proposed method outperformed the currently available forecasts in four Texas utilities in 85% of the developed models. Moreover, large errors in the forecasts of exogenous variables do not result in large errors in forecasts of energy sales or peak load demand, as it is observed in linear forecasting methods.

An inherent characteristic of long-term load forecasting is the uncertainty in the identified model structure, model parameters, as well as future projection of different

exogenous variables. An uncertainty assessment framework is proposed based on statistical bootstrapping methods. The bootstrapping method does not place any restriction on the probability distribution of the forecasting model residuals. The proposed uncertainty method is used to compute the variability of the energy sales or peak load demand. This becomes a measure of uncertainty in the forecasting model structure, parameters and future values of the exogenous variables.

While the proposed forecasting and uncertainty assessment techniques generally result in lower forecast bias, no guarantee of narrower forecast variance can be given. That is, no well-defined relation between the forecast bias and forecast variance was observed. Using the developed uncertainty assessment framework, an assessment of uncertainty in energy sales in different classes of customers and peak load demand in the City of Bryan, Texas was performed.

“Verily the knowledge of the Hour (i.e. Day of Judgment) is with God (alone). It is He who sends down rain, and He who knows what is in the womb. Nor does anyone know what it is that he will earn tomorrow. Nor does anyone know in what land he is to die. Verily with God is full knowledge and He is acquainted (with all things).”

Quran, 31:34

To my parents,

Mr. Akbar Oufi

and

Ms. Hakime Ghazizadeh

ACKNOWLEDGMENTS

I would like to express my most sincere gratitude to my graduate studies advisors Dr. Alton D. Patton, and Dr. Alexander G. Parlos, for their tireless support and encouragement throughout the course of this research. I was very fortunate to have worked under their supervision and witness their dedication to research as well as their students. I would also like to thank my other committee members, Dr. Chanan Singh, Dr. Ali Abur and Dr. Nasser Kentarnavaz for their support and advice.

I am grateful to Mr. Hassan Charara for his invaluable advice during last several years.

I would also like to thank my colleagues, Sunil Menon, Omar Rais, Rube Williams and Jay Muthusami for creating a stimulating environment.

I am thankful for wholehearted support from my brothers Kouresh and Sirius Oufi and my uncle Hussein Ali Ghazizadeh and his family during the period of my study.

652

TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION.....	1
I.1	Electric Load Forecasting	3
I.2	Literature Review	6
I.3	Research Contributions	15
I.4	Organization of Dissertation	16
II	AN OVERVIEW OF ECONOMETRIC FORECASTING METHODS	18
II.1	Introduction.....	18
II.2	Single- and Multi- Step-Ahead Forecast.....	20
II.3	System Model	22
II.4	System Identification	41
II.5	Regression Analysis	53
II.6	Chapter Summary	59
III	LONG-TERM LOAD FORECASTING - A NONLINEAR METHOD	61
III.1	Introduction.....	61
III.2	Objective Function Formulation	63
III.3	Monte Carlo Filtering.....	72
III.4	Flow Chart of Monte Carlo Filtering Process.....	75
III.5	Chapter Summary	93
IV	FORECASTING MODEL DEVELOPMENT	95
IV.1	Introduction.....	95
IV.2	Modeling Objectives	95
IV.3	Texas Municipal Power Agency (TMPA)	98
IV.4	City of Bryan Forecasting Models	98
IV.5	TMPA Forecasting Summary	166
IV.6	Chapter Summary	172
V	UNCERTAINTY QUANTIFICATION: RESAMPLING TECHNIQUES.....	175
V.1	Introduction.....	175

V.2	Resampling Methods	176
V.3	Bootstrapping Techniques	177
CHAPTER		Page
V.4	Nonparametric Bootstrap Confidence Interval Computation.....	187
V.5	Effect of Point Estimator on Forecast Uncertainty	190
V.6	Chapter Summary	222
VI	UNCERTAINTY ASSESSMENT OF CITY OF BRYAN SYSTEM.....	224
VI.1	Introduction.....	224
VI.2	Uncertainty Assessment Method.....	225
VI.3	Uncertainty Assessment of Energy Sales in Customer Classes of City of Bryan.....	227
VI.4	Discussion of Forecast Uncertainty Models for City of Bryan....	248
VI.5	Chapter Summary	248
VII	SUMMARY AND CONCLUSIONS.....	250
VII.1	Summary of the Research Study	250
VII.2	Conclusions from the Research Study	261
VII.3	Recommendations for Future Research Directions.....	262
REFERENCES	263
VITA	269

LIST OF TABLES

TABLE		Page
4.1	Texas Municipal Power Agency Member Utilities Sales and Peak Load Demand (for FY 1993).....	99
4.2	City Residential Customers Time Series for the City of Bryan	101
4.3	Cooling Degree Days (CDDs) Time Series for the City of Bryan	103
4.4	City Residential Adjusted Price of Electricity Time Series for the City of Bryan	105
4.5	City of Bryan City Residential NN-1 Forecasting Model Specifications .	107
4.6	City of Bryan City Residential NN-2 Forecasting Model Specifications .	108
4.7	Unconditional Energy Sales Forecasts and Forecasting Errors of the City Residential Class for the City of Bryan.....	111
4.8	Conditional Energy Sales Forecasts and Forecasting Errors of the City Residential Class for the City of Bryan.....	114
4.9	City Commercial Customers Time Series for the City of Bryan.....	117
4.10	City Commercial Adjusted Price of Electricity Time Series for the City of Bryan.....	119
4.11	City of Bryan City Commercial NN-1 Forecasting Model Specifications	120
4.12	City of Bryan City Commercial NN-2 Forecasting Model Specifications	121
4.13	Unconditional Energy Sales Forecasts and Forecasting Errors of the City Commercial Class for the City of Bryan	124
4.14	Conditional Energy Sales Forecasts and Forecasting Errors of the City Commercial Class for the City of Bryan.....	127
4.15	Rural Residential Customers Time Series for the City of Bryan.....	130

4.16	Rural Residential Adjusted Price of Electricity Time Series for the City of Bryan	132
TABLE		Page
4.17	City of Bryan Rural Residential NN-1 Forecasting Model Specifications	133
4.18	City of Bryan Rural Residential NN-2 Forecasting Model Specifications	134
4.19	Unconditional Energy Sales Forecasts and Forecasting Errors of the Rural Residential Class for the City of Bryan	138
4.20	Conditional Energy Sales Forecasts and Forecasting Errors of the Rural Residential Class for the City of Bryan.....	140
4.21	Rural Commercial Customers Time Series for the City of Bryan.....	143
4.22	Rural Commercial Adjusted Price of Electricity Time Series for the City of Bryan	145
4.23	City of Bryan Rural Commercial NN-1 Forecasting Model Specifications	146
4.24	City of Bryan Rural Commercial NN-2 Forecasting Model Specifications	147
4.25	Unconditional Energy Sales Forecasts and Forecasting Errors of the Rural Commercial Class for the City of Bryan.....	150
4.26	Conditional Energy Sales Forecasts and Forecasting Errors of the Rural Commercial Class for the City of Bryan.....	153
4.27	City of Bryan Peak Load Demand NN-1 Forecasting Model Specifications	155
4.28	City of Bryan Peak Load Demand NN-2 Forecasting Model Specifications	156
4.29	Unconditional Peak Load Demand Forecasts and Forecast Error for the City of Bryan.....	159
4.30	Conditional Peak Load Demand Forecasts and Forecast Error for the City of Bryan.....	162

4.31	Conditional Forecasts of Total System Demand for the City of Bryan....	165
4.32	Summary of Unconditional Energy Sales Forecasts by Customer Class...	168

TABLE		Page
4.33	Summary of Unconditional Total System Peak Load Demand Forecasts .	169
4.34	Summary of Conditional Energy Sales Forecasts by Customer Class.....	170
4.35	Summary of Conditional Total System Peak Load Demand Forecasts	171
5.1	The Average LAST and GPA at 15 American Law Schools, at 1973 [24], [25]	180
5.2	Bootstrap Estimate of Standard Error for Different Replications	184
5.3	Medium Forecast and 95% Confidence Bands of City Residential Customers	194
5.4	City CDDs Time Series for the City of Bryan.....	196
5.5	Medium Forecast and 95% Confidence Bands of City Residential Price (c/Kwh)	198
5.6	Forecast of City Residential Sales , Forecast Error and Lower and Upper Limit 95% Confidence Bands in MWH Using Regression Point Forecaster	203
5.7:	Forecast of City Residential Sales, Forecast Error and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Linear Regression Point Forecaster	208
5.8:	Forecast of City Residential Sales , Forecast Error and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Neural Network Point Forecaster	219
6.1	Forecast of City Commercial Class Sales, Forecast Error and 95% Lower and Upper Limit Confidence Bands in MWH.....	231
6.2	Forecast of Rural Residential Class Sales, Forecast Error and 95% and Upper Limit Confidence Bands in MWH.....	236

6.3	Forecast of Rural Commercial Class Sales, Forecast Error and 95% Lower and Upper Limit Confidence Bands in MWH.....	241
6.4	Forecast of City of Bryan Total System Demand, Forecast Error and 95% Lower and Upper Confidence Bands in MWH.....	244
TABLE		Page
6.5	Forecast of City of Bryan Peak Load Demand, Forecast Error and 95% Lower and Upper Confidence Bands in MW	247

TABLE

LIST OF FIGURES

FIGURE		Page
2.1	Multi-Step-Ahead Prediction Schematic.....	21
2.2	Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARX Predictors	27
2.3	Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARMAX Predictors.....	30
2.4	Block Diagrams of Single-Step-Ahead and Multi-Step-Ahead NARX Predictors	34
2.5	An FMLP Neural Network Architecture.....	37
2.6	A Neural Network Predictor Representation	39
2.7	Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead FMLP Neural Networks (NN) Predictors.....	42
3.1	Different Time Horizons in Developing a Forecasting Model.....	63
3.2	Behavior of the Training and the Validation Set Error Profile When the Stopping Criterion Is As in Equation (3.12): a) Underfitting b) Not Crossing.....	67
3.3	Behavior of Training and Validation Set Error Profile When the Stopping Criterion is as in Equation (3.13)	69
3.4	Behavior of Training and Validation Set Error Profile When Error Profiles Do Not Cross	71
3.5	Behavior of Training and Validation Set Error Profile When Profiles Cross in More Than One Error Level	71
3.6	Flow Chart of Long-Term Load Forecasting System.....	78
3.7	Flow Chart of Module <i>Main</i>	85

659

3.8	Flow Chart of Module <i>Pillar</i>	86
FIGURE		Page
3.9	Flow Chart of Module <i>Decision</i> to Process the Validation Set	89
3.10	Flow Chart of Module <i>Sorting</i> to Sort the Results'.....	90
3.11	Block Diagram of the Long-Term Load Forecasting System	94
4.1	City Residential Customers Time Series for the City of Bryan	100
4.2	Cooling Degree Days (CDDs) Time Series for the City of Bryan (Monthly Calculation)	102
4.3	Cooling Degree Days (CDDs) Time Series for the City of Bryan (Daily Calculation)	102
4.4	City Residential Adjusted Price of Electricity Time Series for the City of Bryan	104
4.5	Unconditional Forecast of the City Residential Energy Sales for the City of Bryan	110
4.6	Unconditional Forecast Errors of the City Residential Energy Sales for the City of Bryan	110
4.7	Conditional Forecast of the City Residential Energy Sales for the City of Bryan.....	113
4.8	Conditional Forecast Errors of the City Residential Energy Sales for the City of Bryan.....	113
4.9	City Commercial Customers Time Series for the City of Bryan.....	116
4.10	City Commercial Adjusted Price of Electricity Time Series for the City of Bryan.....	118
4.11	Unconditional Forecast of the City Commercial Energy Sales for the City of Bryan.....	123
4.12	Unconditional Forecast Errors of the City Commercial Energy Sales for the City of Bryan.....	123

659

660

4.13	Conditional Forecast of the City Commercial Energy Sales for the City of Bryan	126
------	--	-----

FIGURE

Page

4.14	Conditional Forecast Errors of the City Commercial Energy Sales for the City of Bryan.....	126
4.15	Rural Residential Customers Time Series for the City of Bryan.....	129
4.16	Rural Residential Adjusted Price of Electricity Time Series for the City of Bryan.....	131
4.17	Unconditional Forecast of the Rural Residential Energy Sales for the City of Bryan.....	137
4.18	Unconditional Forecast Errors of the Rural Residential Energy Sales for the City of Bryan.....	137
4.19	Conditional Forecast of the Rural Residential Energy Sales for the City of Bryan.....	139
4.20	Conditional Forecast Errors of the Rural Residential Energy Sales for the City of Bryan.....	139
4.21	Rural Commercial Customers Time Series for the City of Bryan.....	142
4.22	Rural Commercial Adjusted Price of Electricity Time Series for the City of Bryan.....	144
4.23	Unconditional Forecast of the Rural Commercial Energy Sales for the City of Bryan.....	149
4.24	Unconditional Forecast Errors of the Rural Commercial Energy Sales for the City of Bryan.....	149
4.25	Conditional Forecast of the Rural Commercial Energy Sales for the City of Bryan.....	152
4.26	Conditional Forecast Errors of the Rural Commercial Energy Sales for the City of Bryan.....	152
4.27	Unconditional Forecast of the Peak Load Demand for the City of Bryan.	158

660

4.28	Unconditional Forecast Error of the Peak Load Demand for the City of Bryan.....	158
------	---	-----

FIGURE		Page
--------	--	------

4.29	Conditional Forecast of the Peak Load Demand for the City of Bryan.....	161
4.30	Conditional Forecast Error of the Peak Load Demand for the City of Bryan.....	161
4.31	Conditional Forecast of System Total Demand for the City of Bryan.....	164
4.32	Conditional Forecast Error of System Total Demand for the City of Bryan.....	164
5.1	Schematic Illustration of Bootstrap [27].....	179
5.2	A Plot of Law School Data Given in Table 5.1 [24], [25]	180
5.3	The Normal Theory Density Function Drawn From a Bivariate Normal Distribution With True Correlation $\hat{\rho}=0.776$ [24], [25].....	182
5.4	Histogram of N=1000 Bootstrap Replications of $\hat{\rho}^*$ for the Law School Data.....	184
5.5	Forecast of the Residential Customer and 95% Confidence Bands	193
5.6	Forecast of City of Bryan Cooling Degree Days.....	195
5.7	Forecast of City Residential Price (cents per Kwh), and 95% Confidence Bands.....	197
5.8	Flow Chart of Uncertainty Assessment Using Regression Point Forecaster.....	200
5.9	Forecast of City Residential Sales and Lower and Upper Limit 95% Confidence Bands in MWH Using Regression Model Point Forecaster ...	202
5.10	Flow Chart of the Uncertainty Assessment Method Using Regularized Linear Regression Point Forecaster.....	205

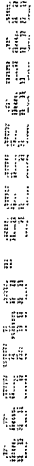
[illegible]

FIGURE	Page
--------	------

5.13	Forecast of City Residential Sales and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Neural Network Point Forecaster.....	218
6.1	Forecast of the City Commercial Customers and 95% Confidence Bands.....	228
6.2	Forecast of City Commercial Price (cents per KWh) and 95% Confidence Bands.....	229
6.3	Forecast of City Commercial Sales and Lower and Upper 95% Confidence Bands in MWH.....	230
6.4	Forecast of the Rural Residential Customers and 95% Confidence Bands.....	233
6.5	Forecast of Rural Residential Price (cents per KWh) and 95% Confidence Bands.....	234
6.6	Forecast of Rural Residential Sales and Lower and Upper 95% Confidence Bands in MWH.....	235
6.7	Forecast of the Rural Commercial Customers and 95% Confidence Bands.....	238
6.8	Forecast of Rural Commercial Price (cents per KWh) and 95% Confidence Bands.....	239
6.9	Forecast of Rural Commercial Sales and Lower and Upper 95% Confidence Bands in MWH.....	240
6.10	Forecast of City of Bryan Total System Demand (MWH) and 95% Confidence Bands.....	243

663

6.11	Forecast of City of Bryan Peak Load Demand (MW) and 95% Confidence Bands.....	246
------	---	-----



CHAPTER I

INTRODUCTION

The primary objective of power system planning is to meet the electrical energy needs of customers as economically as possible with an acceptable degree of safety, reliability and quality. System planning involves studies to achieve this objective in an environment which faces increasing competition as well as increased environmental and regulatory constraints.

For decades before the 1970's, electric utilities operated in an environment that stabilized along a number of lines: cost of fuels was constant, economies of scale dictated ever larger power plants, load growth of about 7% annually was normal, inflation was low, and minimal consumer cost was the utilities' main concern. In such an environment, strategic plans involved increasing the degree of interconnection, selling as much electrical energy as possible and building transmission and generation facilities ahead of load growth. The planning simply involved choosing among several alternatives, where and when to build the facilities.

From the 1970's to 1980's, new challenges faced the utility industry. For the first time, utilities were faced with significant uncertainties in their futures. These uncertainties included uncertainty in load growth, facility siting, regulatory delay and construction lead time, environmental regulations and fuel supply, and more recently, loss of load to competitors. Load growth reduced on the average from moderate in some regions to low or zero in other regions. Social disenchantment with utility plants and transmission lines resulted in stricter state and local approvals for facility siting. Prior environmental approvals were challenged in the courts, and more emission standards

This dissertation follows a style based on the *IEEE Transactions on Automatic Control*.

were introduced. Cartels, foreign nation embargoes, and political disruptions, generated increasing price instability. These trends forced the utility planner to consider new alternatives for dealing with the ever-increasing uncertain future. Several of these alternatives were: continue to use the conventional oil-fired plants or convert to cheaper alternatives such as coal-oil mixture, continued or ceased use of nuclear energy, load management techniques, reduced load through conservation, bulk purchase of energy from outside utility producers, and use of renewable resources.

The 1990's has brought the wave of deregulation. This is supposed to break down the regulated monopolies which have governed the electric power industry for decades, and be in line with the public desire for reduced energy prices. As a consequence, electricity would be traded as a commodity, with the consumer having the ability to shop around for the cheaper supplier. As a further implication of this policy, the generation system, the transmission system and the distribution system within a parent vertically integrated utility might be unbundled, and eventually owned by different owners. The owner of the transmission system should allow the passage of energy among different suppliers and customers, charging them the cost of "transmission wheeling". In this new scenario, not only the total amount of future energy demand is uncertain, but the percentage share of the total demand which would be bought from a particular supplier is also unknown. The level of future uncertainty is so high that utility planners have reduced their long-term planning horizon to just a few years into the future. In this competitive market, utilities can no longer charge the consumers for their inaccurate or possibly wrong decisions and investments. With the reality of overwhelming uncertainty in the future, utilities are still forced to develop long-range plans. Therefore, more accurate and sophisticated planning tools should be developed to cope with the new challenges and constraints.

The first step in the planning process is forecasting the long-term energy and peak load demand in the service area. Different components of the power system planning process are based directly on the results of the long-term load forecasts.

Generation capacity expansion programs use the predicted annual energy and peak load demand to assess the need to build new generating units [70]. In system reliability studies, the major reliability indices which are the probability of the system being able to meet the peak demand and the probability of the system being able to meet the hourly load demand are calculated based on the load forecasts [10]. Transmission system expansion plans are directly based on the outcome of the system and distribution level long-term load forecasts. The demand growth in the distribution system is not uniform and areas of high or low growth may exist within a region [75].

I.1 Electric Load Forecasting

Load forecasting in a system falls into four categories:

1- Very short-term load forecast: extending to 30 minutes into the future in real time. This forecast is generally required in Automatic Generation Control (AGC) to regulate the frequency in multi-area interconnected power systems, and economically distribute generation among the available resources. Mathematically, this is a tracking problem in which load is forecasted for the next 30 minutes based on the past 24 hours of load data. This prediction is used to dynamically dispatch each available generator [44].

2- Short-term load forecast: extending from one hour to one week into the future. This forecast is generally required for scheduling functions such as hydro-thermal coordination, economic dispatch, unit commitment and interchange transaction scheduling and load management. Mathematically, based on the daily load and temperature information of the past few years, the daily peak demand is forecasted for up to one week [49].

3- Medium-term load forecast: extending from one month to one year. This forecast is generally required for maintenance scheduling of generating plants and the transmission system. Mathematically, based on the past several years of monthly

historical peak load, energy demand, temperature, and social and economic indicators, the monthly energy and peak load demand is forecasted for up to a year [7].

4- Long-term energy and peak load demand forecast: extending from one year to ten years. This forecast is generally required for planning system generation and transmission resources. Mathematically, based on available historical annual data of peak load, energy demand, population, temperature, electricity prices and relevant economic indicators, the annual energy and peak load demand are forecasted for up to ten years into future [29], [57], [58].

The future horizon times in each of the above four forecast categories results in strong distinctions in methods used for forecasting in each of the categories. In this study, we will exclusively concentrate on techniques developed for forecasting long-term energy and peak load demand.

A long-term load forecasting system relates social and economic variables at the local, state and national level to the electricity demand in the service area. In general, the consumers in a service area are classified as residential, commercial and industrial customers. The load of each class is particularly related to certain variables. For example temperature affects the consumption patterns of residential customers more than industrial customers. Currently, changes in economic activities tend to affect industrial customers more than residential customers. It is not uncommon to observe structural changes in the consumption patterns of a particular class of customers over a period of time. Thus, explaining variables must be adjusted over time.

In long-term forecasting, historical information tends to be sparse. In general, the historical data base includes annual data of different influential variables for, at best the past 25 years. This generates approximately 25 distinct data points. However, a fraction of this data set should be reserved for test and validation of the developed model [45]. As a result, a good portion of these points will not be directly used in the model development process. Moreover, data collected some 25 years ago can not be relied upon

strongly. In many small electric utilities, relevant historical data has not been recorded in a systematic fashion. This often forced the forecaster to replace the unavailable information by data recorded in the neighboring city or county with associated inaccuracies. The introduction of computers in the early 1980's improved the data collection techniques in electric utilities and resulted in more accurate and complete data base systems [13].

Small and unreliable historical data set tended to force utility forecasters to avoid sophisticated modeling techniques. In general, a simple linear model was often used to model and forecast the energy demand [13], [22]. However, the stable energy markets of the past tended to forgive the gross errors in the models developed based on such unreliable data bases. However, from the early 1980's, when the electric utility industry faced increased uncertainty, the quality of outcomes from such models deteriorated and the utilities adopted more sophisticated forecasting techniques. Currently, long-term forecasting systems includes tens of forecasting submodels, each forecasting an exogenous variable, such as fuel price, operational and maintenance cost [57], [58].

As many variables involved in electric load forecasting must be accurately predicted into the future, an integrated forecasting system must account for uncertainties in predicted energy demand resulting from uncertainties in the predicted values of exogenous variables (or indicators) as well as from uncertainty in the selected model structure and the computed model parameters. If the forecasted demand is very uncertain, then the risk associated with generation options which have long lead-times and high fixed costs but low variable costs may be too great. Generation capacity options with shorter lead-times and lower fixed costs may become desirable even though these might have higher operating costs. With the current overwhelming uncertainty in the utility industry environment, the later option is the general trend in the industry. The uncertainty associated with future demand can make the system facilities inadequate and risky or excessive and uneconomical, both being unacceptable. The uncertainty associated with capital costs and fuel prices can make an initially sound economic

decision a poor decision after the facility is built because of several years of time lag between the decision and completion of the project.

The resource planning, in general, and load forecasting, in particular, are subject to many forecasted parameters such as fuel prices, plant cost, technology availability, customer behavior, environmental and regulatory requirements [63]. Expert opinion clearly indicates that a major source of uncertainty in future planning requirements, and operation of existing generation resources is the forecasted load demand [52]. Due to the long future horizon, any realistic long-term load forecast must account for the inherent uncertainties in future projections [36].

I.2 Literature Review

The first published work on long-term load forecasting dates back to 1946, when a time-trending method was used to forecast the energy demand at Detroit Edison Company [64]. Since then, long-term load forecasting has been an area of continuous attention both in the power systems community, as well as in the economics community. Several factors have contributed to the increasing sophistication in long-term load forecasting methods employed by electric utilities. Among them are the drive towards higher accuracy and ever more uncertain socio-economic environment. Currently, due to the large amount of information required to initiate a useful forecasting program for understanding customer behavior, as well as in designing rate structures, a department in every electric utility is involved with the forecasting process [57], [58].

I.2.1 Long-Term Load Forecasting Methods

Forecasting models have evolved through the years from a single equation model [71] to complex systems, consisting of different subsystems, each modeling one of drivers of the energy demand [57], [58]. Major methodologies used in long-term load forecasting can be categorized as follows [23]:

1- Time-Trending Methods: These methods were mainly used prior to the 1970s due to stable demand trends [64]. In this technique the time-trend of a particular class of customers is obtained and then extrapolated to obtain the forecasts of the future demand. Some of the techniques utilize the models based on transformations of load data, which are then used for curve fitting. The assumption in using these conversions is that a transformation of data produces a simple curve which easily explains the trend in energy consumption. Most commonly, a constant percentage rate of growth were adopted. However, in the more complex forms exponential growth curves, saturating growth curves, and three point methods were employed. In the exponential growth curve method a semi-logarithmic plot of data would yield a straight line. The slope and the intercept of the fitted straight line thus defined the parameters of the demand growth trend. The saturating growth curve method tried to more clearly explain several realistic stages in the development of every service territory. These stages consist of an initial period of relatively slow but gradually increasing growth period. This stage is followed by an intermediate period of rapid growth. And finally a period where the rate of the growth declined and the observed load and energy demand appeared to reach a saturation level. Recognizing these facts, the exponential growth methods uses logarithmic functions to fit the historical demand data and forecast the future energy demand. The three-point-method recognizes the fact that the growth rate in a service area differs in different stages of the development. However, it replaces the single nonlinear logarithmic growth function with three linear segments. Each linear segment is identified based on the corresponding data from the service area. More recently, Barakat [1], [2] has reported the application of the decomposition technique in a time-trending framework to forecast the total system peak load demand in a Middle Eastern electric utility with a high growth rate.

In general, the data requirements of time-trending methods are minimal, mainly the historical energy or peak demand. Also, minimum computational capabilities and

expertise is required. However, the time-trending techniques are vulnerable to errors resulting from changes in social and/or economic factors affecting energy demand.

2- Time-Series Methods: These methods estimate the underlying patterns in demand and project a repetition of the historical pattern in the future. They are also commonly known as Box-Jenkins methods. These methods are based on the assumption that there is a strong correlation between successive annual peak loads or annual energy demands [35]. The initial proposed model by Box and Jenkins was AutoRegressive Integrated Moving Average (ARIMA) [35]. This model describes the current value of the ~~peak~~ peak demand in terms of a finite linear function of load in previous years, as well as the difference in the load demand between consecutive years. In practice, simpler versions of this model have also been used in forecasting energy demand. If the current values of energy are described only as a finite linear function of previous values of the load, the model is called an AutoRegressive (AR) model. It is necessary to identify the order and the parameters of the model. If the current value of energy is described only as a finite linear function of previous error values of the load, the model is called a Moving Average (MA) model. The error is the difference between the load demand in two consecutive years. An AutoRegressive Moving Average (ARMA) model includes both of components. In an ARMA model, it is necessary to identify the order and parameters of the AR and MA components. The above models generally assume that the growth in load demand can be described as a stationary process. However, if the data series exhibits a non-stationary behavior, then the series will be differentiated until a stationary series is obtained and the resultant data series is modeled using one of the above treatments. The forecast obtained from the stationary model will be integrated to obtain the forecast of the peak load or the energy demand. The model obtained using this approach is called an AutoRegressive Integrated Moving Average (ARIMA) model [35]. In addition to parameters existing in a stationary process, the number and the form of the differentiation in the ARIMA model must be determined.

The data requirement of a time-series model are minimal, mainly the historical energy or the peak demand [69]. Moderate statistical knowledge is required to identify the class of time-series which best represents the historical trend, and the order and the parameters of the model. The time-series models do not reflect the effect of influencing variables on load demand. They assume that the future socio-economic conditions will be the same as in the past and can not make use of future indicators.

3- End-Use Methods: These methods are mostly applied to the residential class and less frequently to the commercial and the industrial classes, due to the homogeneity of the residential users [46]. They basically sum the energy consumed in end-uses, such as refrigerators and water heaters, considering the number of these appliances and the energy consumed per appliance. The number of a particular house-hold appliance depends on the number of customers in the service area times the percentage of customers having the particular appliance. At the time of invention of a particular appliance a small number of customers find it necessary or economically affordable. This number gradually increases, until it reaches a constant percentage of the total number of customers after several years. This percentage level is generally referred to as the saturation level. This pattern can generally be represented as a logarithmic function with a final saturation level. The consumed energy per appliance is also a nonuniform function. The energy usage per appliance is initially high. However, as the efficiency of the appliance improves, the energy usage per appliance drops to reach a constant level after several years. Estimates and forecasts of such quantities are the principle requirements in every end-use model. Tens of such appliances in each of the residential, industrial and commercial classes should be accounted for. The data requirements in end-use modeling include the historical levels of appliance efficiency, the degree of thermal insulation in residential and commercial buildings and industrial process energy consumption data. Inclusion of use-per-appliance makes the end-use structures amenable to adjustment to reflect the effect of conservation policies. It is obvious that the cost, skill and manpower requirements in end-use modeling is high. Moreover, with the inclusion of such detailed

information about the service area, the key to effective end-use forecasting is still an accurate forecast of the population, the number of customers in the service area, and the saturation level [23]. The inaccuracy in these forecasts are the main sources of inaccuracy in end-use models. End-use model do not provide any means to study the effect of electricity prices on the consumption pattern. A trend has been noticed towards an increase in the use of econometric methods to develop forecast estimates of such data.

4- Econometric Methods: These methods are the most complex forecasting methods available, but utilities are increasingly acquiring the necessary expertise and experience in efficiently using them. The method is based on more fundamental economic analysis and came into prominence during the early sixties. This reflected the growing sophistication of applied economic analysis and data. In principle, these methods estimate the functions relating electricity usage to underlying factors (indicators). The required data include historical observations of the number of customers in particular class, customer income, commercial and industrial activities, the price of electricity and alternative fuels and weather attributes. Estimating model parameters requires a fair knowledge of statistics and econometrics. The simplest econometric models use three basic service classes: residential, commercial and industrial plus a residual class [13]. In current utility practice, the form of the econometric equations can be linear or log-linear, depending on ease of interpretation or accurate representation of the underlying relationships. In econometric modeling, the variables are divided into different categories as driving (exogenous) variables, intermediate variables and endogenous variables. It is possible to model lagged variables to account for variables which require passage of time to impact electricity usage. Forecasting energy demand requires that forecasts of exogenous variables be available. Errors in forecasting exogenous variables are obviously a major source of errors in econometric forecasts. Econometric models range from simple models with three or four equations [13], to elaborate multi-equation models estimated by sophisticated statistical techniques [57], [58]. Recently, Liu [43] has reported the application of Artificial Neural Networks (ANN) in an econometric framework, and

compared the results with that of a multiple linear regression (MLR) model. Without using any regularization or network optimization, they concluded that the MLR forecasts outperform those of ANN. Another application of ANN to long-term load forecasting in the econometric framework has been reported by Tummala and Fung [72]. Without considering the unique characteristics of long-term load forecasting data set, they concluded that ANN performance is at least as good as that of the MLR.

The econometric methods are often the only available methods to measure the effects of price, economic activities, and weather on electricity demand. The advantages of econometric modeling can be summarized as providing estimates of the impact of underlying factors, tracing sources of forecast errors and forecasting service area economic and demographic variables. However, developing such models requires high skill and experience. Moreover, some of the exogenous variables, such as fuel cost or income, are difficult to accurately forecast.

5- Hybrid End-Use and Econometric Methods: The traditional approach to forecasting involves choosing the forecasting method judged most appropriate of the methods and applying it to some specific situation. The rationale behind such an approach is the notion that a best method exists and can be identified. However, in practice situations arise where more than one model performs adequately on the available historical data. Moreover, as presented in the previous sections, different forecasting methods have particular advantages in identifying and utilizing one component of the overall load forecasting problem. It has been shown that the combination of several suitable candidates results in more accurate forecasts than forecasts obtained from a single model [48]. Combination forecasts can be performed in two forms. It is possible to use two or more methods to estimate and forecast different components of a forecasting system and then combine these results to arrive at a single forecast. McMenamin [46] reported an application where econometric models were used to forecast different indicators in an end-use framework to forecast the sales in a class of residential customers. It is also possible to use different models independently to forecast the

energy demand, and obtain several forecasts of the future energy demand. The final forecast will be a combination of these independent forecasts. Several techniques have been reported which combine the outcome of different forecasting models. The simple average of forecasts from different independent models has shown to be in general more accurate than individual forecasts [68]. Newbold and Granger [48] combined different independent forecasts, based on the covariance matrix of forecast errors.

The required data in hybrid models includes the combined data requirements of the original model structures. The hybrid methods require high statistical and programming skills. Their development and maintenance also requires large effort. Several commercial forecasting packages such as REEP, COMMEND and ORNL use hybrid modeling techniques [23].

1.2.2 Uncertainty Assessment in Long-Term Load Forecasting

The experience of the 1970s and 1980s has clearly shown that the future demand for electricity is highly uncertain. Even with the profound advancements in forecasting methods, we have to recognize the validity of the general rule that: *The Forecast Is Always Wrong!* [63]. Understanding the uncertainty of point forecasts of demand enables a utility to prepare alternative plans to face a future other than the most expected one. Using a flexible planning approach, for example, Baltimore Gas and Electric postponed construction of two 640 MW generating units for 20 years [21]. Two published reports by the Electric Power Research Institute (EPRI) have summarized the utility practice with forecasting under uncertainty [28], [29]. Long-term demand forecasting involves three main types of uncertainty.

- 1- All Pervasive: The forecaster does not know any structural relationship between the variables at hand and the demand. That is, there is no information about the variables, and how these variables may be related to each other and

ultimately to electricity demand. An example of such uncertainty is, the effect of a breakthrough in electric vehicle technology on electricity demand.

- 2- **Uncertain Model Structure:** The forecaster has some idea about the important factors in the analysis, and how they are generally related to each other and to the demand for electricity. However, there are several alternative model structures for explaining the relationship between the factor at hand and the demand for electricity, and the uncertainty refers to which of these model structures is the most appropriate. Additionally, the forecaster does not usually know with certainty the future values of the exogenous variables or the values of the model parameters.
- 3- **Uncertain Model Parameters and Inputs:** The forecaster has a credible model for electricity demand. The uncertainty results from the future values of the exogenous variables, the values of model parameters and general uncertainty resulting from the random errors present in all model relationships. This is the simplest type of uncertainty to deal with, and there is a vast amount of literature regarding its treatment.

Several methods have been developed to deal with the uncertainty. These methods range from analytical techniques with minimum computational requirements to elaborate simulation techniques. The required computational efforts increase, as more realistic assumptions about the nature of uncertainty are adopted. The uncertainty analysis methods can be categorized as follows:

Scenario Planning Methods: These methods are generally considered to be the most appropriate in dealing with problems characterized by *all-pervasive* uncertainty. A scenario is a hypothetical sequence of events constructed for the purpose of focusing the attention of the planner and the decision maker on causal processes and decision points. Scenarios can effectively organize a variety of seemingly unrelated information dealing with economic, technological, competitive and social variables into alternative worlds and

enable the utility planner to prepare the appropriate future responses for different future candidate [41].

Analytical Methods: These methods are generally used in the simplest forecasting models. For example, a one equation econometric model obtained using linear regression analysis, with the additional assumption of normal error distribution and no uncertainty in exogenous variables [55].

Simulation Methods: These methods can be either *deterministic simulation techniques* or *stochastic simulation techniques*. In *deterministic simulation*, several scenarios of input projections are made as low, medium and high, based on personal judgment and expert opinion. Different projections of input variables within a scenario should be reasonable. These scenarios can be developed for forecast models using econometric or end-use methods. *Stochastic Simulation Techniques* are computationally intensive. The Monte Carlo Method is an example of stochastic simulation. Simple Monte Carlo methods are based on random samplings of probability distributions of model parameters and inputs. The uncertainty in a model parameter is represented by a normal distribution with the mean equal to the estimated value and standard deviation equal to the standard error of the coefficient. In principal, the uncertainty in the exogenous variables is also represented as a distribution around the base forecast. However, based on the expert opinion, it might have an unsymmetrical distribution [62], [29], [40]. Using the distribution of the exogenous variables and model parameters, a distribution of the forecasted demand for the forecasting period is developed. Different confidence levels on the demand forecast are computed using the demand distribution. Irish [39] used a discrete approximation to the Monte Carlo approach, called Probability Tree Method, to compute load forecasting uncertainty. Discrete distributions were specified for all the uncertain parameters and input variables, as opposed to continuous distributions. This was in the form of high, low and medium values with the associated probabilities for the inputs. The interrelationship between the parameters and the input variables were defined through a tree diagram. The model was simulated for each branch

of the tree, and the simulated output was assigned a probability equal to the probability of occurrence of the branch *i.e.* the joint probability of occurrence of all the outcomes of that branch.

Bernard and Veall [6] published the only application of Bootstrapping simulation techniques to long-term load forecasting. The bootstrapping method which was developed by Efron [24] belongs to the class of stochastic simulation methods of uncertainty quantification called resampling plans. Bootstrapping basically uses the initial model residuals to build an ensemble of model parameters. Using this ensemble, called the empirical distribution, the uncertainty in model parameters as a measure of the uncertainty in the model can be evaluated. Bernard and Veall [6] quantified uncertainty in the peak demand by using data from Hydro Quebec for the single year of 1990. A two equation model comprising of an equation relating the peak demand to the average demand, and an equation relating the average demand to an electric price variable and an economic activity variable were estimated. The bootstrap was performed in three stages, removing assumptions regarding the uncertainty of the forecasted exogenous variables in consecutive stages. The average of the forecasted peak in the year 1990 remained almost the same in the three scenarios, while the uncertainty in the estimate increased as more of the unrealistic assumptions were removed. To better model the tails of the empirical distribution of residuals, the *non-parametric bootstrap* was replaced by a *parametric bootstrap*, using a convolution of a normal distribution and the empirical distribution of the residuals. The results were exactly identical to those of *non-smoothed bootstrap*.

I.3 Research Contributions

The main contribution of this research is to develop a comprehensive long-term load forecasting system, which includes an accurate and robust long-term load forecaster and an uncertainty assessment technique. The research aims at providing the utility planner with the state-of-art in forecasting techniques.

In particular, as a result of the research described in this dissertation, the following contributions have been made in the area of long-term load forecasting:

1. Development of regularization techniques suitable for data scarce forecasting problems. This development utilizes recent advances in the field of Artificial Neural Networks as tools for nonlinear system identification.
2. Application of recent advances in the field of nonparametric statistics in particular bootstrapping, to develop a systematic and reliable method for uncertainty assessment. Uncertainty in forecast model structures, parameters, and future values of indicators are accounted when computing the confidence level of the energy sales and peak load demand forecasts.

1.4 Organization of Dissertation

In Chapter II an overview of linear and nonlinear system models, system identification and econometric forecasting methods in an input/output framework are presented. The concepts of Single-Step-Ahead forecast and Multi-Step-Ahead forecast are introduced. Linear and nonlinear system descriptions are reviewed. The System Identification section reviews the empirical modeling of linear and nonlinear systems in an input/output setting. The Neural Networks (NN) as a nonlinear system modeling tools and their expansions are introduced. Multi-Step-Ahead prediction methods using NN are described. Linear and nonlinear regression analysis are also reviewed.

In Chapter III the application of nonlinear forecasting methods to electric load forecasting along with the contribution of this dissertation in the point forecast computation are presented. The selection of the objective function, regularization against overfitting and formulation of the multi-objective optimization problem are presented. The application of the Monte Carlo filtering technique in the selection of the architecture and parameters of the forecasting model is presented. Several stopping criteria are

introduced and advantages and drawbacks of each criteria in the filtering process are discussed.

In Chapter IV applications of several nonlinear forecasting methods developed in this dissertation to forecast the energy sales, total system and peak load demand in four Texas electric utilities are presented. Unconditional forecasts of energy sales for different classes of customers are performed using actual observations of different input variables in energy sales models. Moreover, using the projection of different input variables in energy sales models, conditional forecasts of energy sales are also performed. Results of several of the models developed are compared to results obtained by the participating electric utilities using alternative models.

In Chapter V the statistical resampling techniques for uncertainty quantification are described along with the contribution of this dissertation in uncertainty assessment of long-term load forecasts. The bootstrapping techniques are introduced and their applications in model, parameter and prediction uncertainty assessment are described. Several methods for nonparametric computation of forecasting confidence intervals are described. A comparison is performed on the confidence band developed when the point forecast estimator is a linear model, regularized linear model and regularized nonlinear model.

In Chapter VI the application of the developed uncertainty assessment method to different classes of customers in the City of Bryan Utilities is presented. Based on expert opinion an uncertainty band around the projection of each of input variables is formed. The form of the band depends on the nature of uncertainty in the particular variable. The bootstrap technique utilizes the residuals from the original developed forecasting model to generate an ensemble of historical observations. This ensemble is used to assess the model parameter uncertainty. The developed confidence bands assess the uncertainty in the energy sales due to model parameter as well as conditional uncertainties.

A summary of this dissertation, the conclusions reached from this research, and directions for further research are given in Chapter VII.

Digitized by Google

CHAPTER II

AN OVERVIEW OF ECONOMETRIC FORECASTING METHODS

II.1 Introduction

Methods for load forecasting have changed during the last two decades. In the period prior to 1970's, time-trending and time series methods sufficed to explain and forecast the growth in energy demand. Since the 1970's overwhelming uncertainties necessitated the development and application of more sophisticated forecasting techniques, among them end-use and econometric techniques.

The underlying principle in econometric forecasting is identifying a cause and effect relationship between energy demand and economic, social, and geographic variables [55], [23]. Among econometric variables are the price of electricity and alternative fuels, Consumer Price Index (CPI) and Gross National Product (GNP). Social variables include the number of customers in the service area, total population, net immigration in flow, out flow and per capita income. Among geographic indicators are the average, maximum and minimum annual temperature, and heating and cooling degree days. The cause-and-effect relationship explains changes in load demand by relating to changes in the relevant indicators. A forecast of energy demand is based on the forecast of the relevant variables.

From a System Identification (SI) prospective, econometric forecasts are input-output relationships [45], where the inputs are different relevant social, economic and geographic indicators, and the output is the energy sales or the peak load demand. Development of an econometric energy forecasting program involves the following major steps [4].

Indicator Identification involves identification of the most influential social, economic and geographic variables affecting the energy demand in the service area. The correlation between candidate influential variables and the energy demand should be computed to determine the influential variables. A correlation coefficient close to one indicates a strong positive relationship. A correlation coefficient close to minus one indicates a strong negative correlation while zero indicates negligible correlation. The indicator identification process should account for any structural changes within the historical data set. In these situations the cause-and-effect relationship between a particular indicator and the energy demand changes or ceases to exist over time. One approach to detect such structural changes is to compute the correlation between the exogenous and endogenous variables in different historical periods and compare their values.

Model selection involves selection of a model structure to relate the selected influential variables with energy demand. The model structure can be linear or nonlinear [4]. It can include lags of the influential variables and/or the energy demand [53]. In general, the selection of a structure is based on expert judgment. In the absence of such knowledge, different candidate structures may be iteratively tested on the available data set. Due to the scarcity of historical information in long-term load forecasting, it is common to find more than one model structure performing satisfactorily on the historical data set. It has been shown that combining the forecasts from different model structures, results in more accurate forecasts [68].

Parameter Estimation involves determining the parameters of the selected model structure which relates different influencing variables to energy demand. The parameter estimation method depends on the model structure selected in the previous step. The linear and nonlinear estimation techniques are quite distinct. The estimation techniques

can involve either a single or multiple iterations [45]. Due to the scarcity of the historical information, it is easy to develop simple models. However, most simple models fail to extract complete information from the available historical data set [4]. Linear models fail for example, to extract nonlinear relations among temperature and energy demand. Large nonlinear models on the other hand, may introduce unnecessary complexity and increase the difficulty of parameter estimation. The estimation technique should account for such dangers.

In this dissertation, the load forecast is viewed as a system. Similar to any other system, the forecast has its own inputs and outputs. The inputs are different variables affecting the energy demand and the output is the energy sale to a class of customers in the service area. However, similar to any other system, the load forecasting has its own unique problems. These unique problems include the scarcity of data, stochastic nature of the data and uncertainty in the available historical data as well as in the future projections of different involved variables. Recognizing load forecasting as a system allows the forecaster to benefit from a set of proven tools in the area of system engineering in dealing with the problem.

This chapter is organized as follows. Section II.2 explains the concept of Single-Step-Ahead and Multi-Step-Ahead forecast. Section II.3 describes system models. The system models can be linear or nonlinear. The probabilistic and predictor forms of different system models are described. Neural Networks (NN) as nonlinear system predictors will be introduced. Section II.4 describes System Identification (SI) techniques and empirical modeling. Linear and nonlinear estimation techniques are described. The Backpropagation learning algorithm as a nonlinear estimation technique will be reviewed. Section II.5 describes linear and nonlinear regression models and also contains a review of

linear and nonlinear least-squares estimation. Finally, section II.6 summarizes the chapter.

II.2 Single- and Multi- Step-Ahead Forecast

Forecasting or prediction refers to estimation of a variable of interest (such as energy demand) at a future time based on information available up to and including the

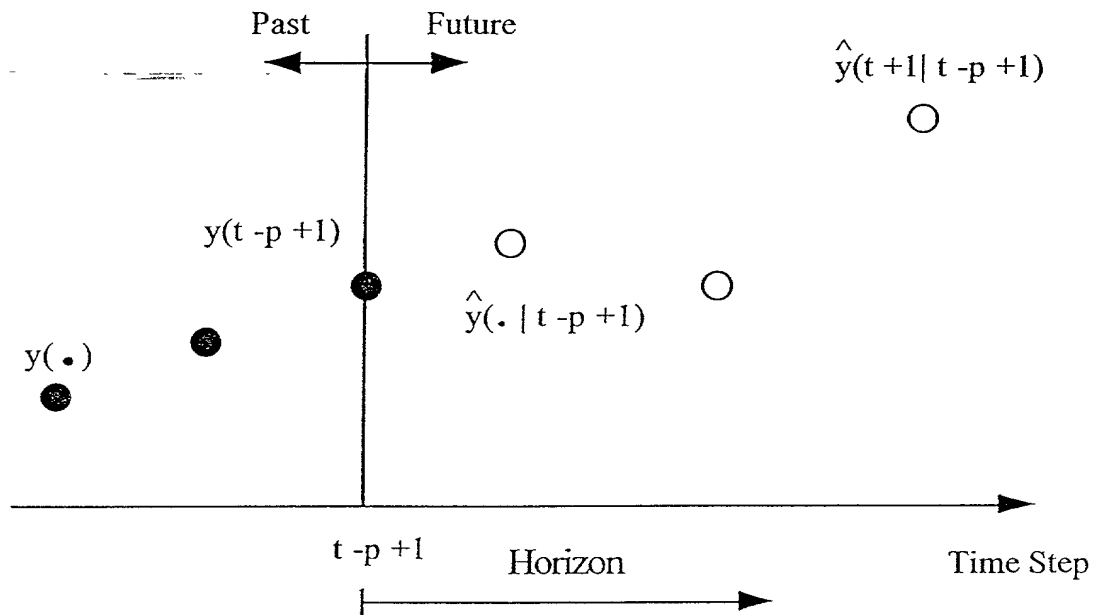


Figure 2.1: Multi-Step-Ahead Prediction Schematic.

current time t . Two types of forecasting problems can be formulated: Single-Step-Ahead forecast also known as SSP (Single-Step-Prediction) and Multi-Step-Ahead forecast, also known as MSP (Multi-Step-Prediction). SSP is the forecast of the variable of interest at time of interest $t+1$, based on the information up to and including time t . In such systems, forecasting the variable at time $t+2$ requires the value of the variable at $t+1$ to

be available. An example of such a system is the forecast of tomorrow's peak load demand.

MSP is the prediction of the variable of interest at time $t+p$, given the information up to and including time instant t . In such systems, forecasting the variable at $t+2$ does not require that the actual value of the variable of interest at $t+1$ be known. An example of such a system is long-term load forecasting where the forecast of energy for several years into future is desired.

Theoretically speaking, in order to perform a MSP, one should directly relate the forecast at time $t+p$, that is $\hat{y}(t+p|t)$ to measurements $y(t-p), y(t-p+1), \dots$, etc (Fig. 2.1). In practice, however, MSP is performed by relating the forecasts $\hat{y}(t+p|t)$ to the immediate previous forecasts, $\hat{y}(t+p-1|t), \dots$. In other words, to perform a single p -step-ahead forecast, we have replaced it by p single-step-ahead forecasts. It has been shown that under simplifying assumptions the optimal mean-square MSP error can be obtained by such a recursive procedure [69]. In other words, to forecast the energy in the year 2007, which is a 10-year-ahead forecast, 10 single-step-ahead forecast should be performed from year 1997 and the forecast of each year should be used in forecasting the next consecutive year.

In the system identification literature, the MSP formulation is presented slightly differently. It is assumed that the earliest available information to be at time instant $t-p+1$, and the last to be $t+1$. Therefore, it is desired to forecast $\hat{y}(t+p+1|t+1)$. It can be easily shown that the above two formulations are identical. In this dissertation the SI approach is followed for consistency with the available literature.

II.3 System Model

In loose terms a *system* is a framework in which variables of different kinds interact and produce observable signals. These observable signals are usually called outputs. The system is also affected by external stimuli. External signals that can manipulate the system are called inputs [45].

The notion of a system is a broad concept and it is not surprising that it plays an important role in modern science. With such a broad definition, most things in our environment become systems: the solar system, a power plant and so on. Considering the above definition of a system, a number of questions as the input and the output of the system arise. To answer such questions, a “model” of the system is desired to investigate and explain the behavior of the system and generate an appropriate observable signal (output) as a result of changes in the stimulus (input). Models can be physical or mathematical [45]. With existing computer facilities and using mathematical formulations, it is possible to “simulate” the behavior of systems and study their response (*output*) under various stimuli. However, the accuracy of simulation depends on the quality of the system model.

The system model can be linear or nonlinear. Linear models assume that the system can be represented adequately using linear mathematical relationships, while nonlinear system models assume that the system must be represented using nonlinear mathematical relationships.

The system models can be input-output or state-space models. Input-output models only include measurable input and output quantities. State-space models include inputs, outputs and intermediary descriptors called states. In input/output models, the

observed output signal is represented in terms of current and past inputs and past outputs.

In econometric long-term electric load forecasting the system model is the developed cause-and-effect relationship which relates the input variables including different relevant social, economic and geographic variables to the output which is MWh energy sales or MW peak load demand. This system model is exclusively represented as an input-output model. Based on this, in this dissertation, input/output system models only are discussed.

II.3.1 Linear System Models

Linear system models assume that the system can be adequately represented using linear relationships. The majority of current modeling techniques use linear system models. This is mostly because of the abundance of linear parameter estimation techniques available to identify the linear system model parameters.

Linear input-output system models that are most frequently used are Auto-Regressive with eXogenous inputs (ARX), and Auto-Regressive Moving Average with eXogenous inputs (ARMAX) [45].

II.3.1.1 ARX Models

II.3.1.1.1 Probabilistic Form of ARX

One of the most commonly used linear system models in the Auto-Regressive with eXogenous inputs (ARX) model. A general ARX model for a multi-input multi-output (MIMO) system is represented as:

689

$$\begin{aligned} \mathbf{y}(t+1) = & \mathbf{A}_1 \mathbf{y}(t) + \cdots + \mathbf{A}_{n_y} \mathbf{y}(t - n_y + 1) \\ & + \mathbf{B}_1 \mathbf{u}(t) + \cdots + \mathbf{B}_{n_u} \mathbf{u}(t - n_u + 1) + \mathbf{e}(t), \end{aligned} \quad (2.1)$$

where

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix} \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad \mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ \vdots \\ e_n(t) \end{bmatrix} \quad (2.2)$$

n and m are the number of outputs and inputs respectively, $\mathbf{y}(t)$ is the output vector of the ARX system model, n_y is the number of past outputs (lag terms), $\mathbf{u}(t)$ is the input vector to the ARX system model, n_u is the number of past inputs (lag terms) in the model, and $\mathbf{e}(t)$ is the assumed noise vector with a zero mean and Gaussian distribution. The coefficients of the ARX model, A_i for $i = 1, 2, \dots, n_y$ and B_i , for $i = 1, 2, \dots, n_u$ are assumed to be known.

The general single-input single-output (SISO) ARX system model is represented by the following equation:

$$\begin{aligned} y(t+1) = & a_1 y(t) + \cdots + a_{n_y} y(t - n_y + 1) \\ & + b_1 u(t) + \cdots + b_{n_u} u(t - n_u + 1) + e(t), \end{aligned} \quad (2.3)$$

where $y(t)$ is the output of the ARX system model, n_y is the number of past outputs (lag terms), $u(t)$ is the input to the ARX system model, n_u is the number of past inputs (lag terms) in the model, and $e(t)$ is the assumed noise term. The coefficients of the ARX model, a_i for $i = 1, 2, \dots, n_y$ and b_i for $i = 1, 2, \dots, n_u$ are assumed to be known.

II.3.1.1.2 Predictor Form of ARX

Using an ARX system model in forecasting as well as in parameter estimation requires that the single-step-ahead prediction (SSP) or the multi-step-ahead prediction (MSP) form of the ARX system model be derived from equation (2.1) or (2.3). The SSP or MSP form of an ARX model are also called its *predictor* form [45]. The predictor form of an ARX system $\hat{y}(t+1|t)$ describes the output of the system at time $t+1$, if the system parameters and all its inputs and outputs with their corresponding lags at time t are known.

The SSP output form of the MIMO ARX system model given in equation (2.1) is given by equation:

$$\begin{aligned}\hat{y}(t+1|t) = & A_1 y(t) + \dots + A_{n_y} y(t - n_y + 1) \\ & + B_1 u(t) + \dots + B_{n_u} u(t - n_u + 1),\end{aligned}\quad (2.4)$$

where $y(t)$ and $u(t)$ have been defined in equation (2.2), A_i for $i = 1, 2, \dots, n_y$ is the set of output coefficient vectors and B_i for $i = 1, 2, \dots, n_u$ is the set of input coefficient vectors. Equation (2.4) is also called the SSP predictor form of a MIMO ARX model.

Using the SISO ARX model given by equation (2.3), the corresponding SSP of the system output $\hat{y}(t+1|t)$ is given by equation:

$$\begin{aligned}\hat{y}(t+1|t) = & a_1 y(t) + \dots + a_{n_y} y(t - n_y + 1) \\ & + b_1 u(t) + \dots + b_{n_u} u(t - n_u + 1)\end{aligned}\quad (2.5)$$

where $y(t)$ is the output of the ARX system model, n_y is the number of past outputs (lag terms), $u(t)$ is the input to the ARX system model, n_u is the number of past inputs (lag terms) in the model. Equation (2.5) is also called the SSP predictor form of the SISO ARX model.

The MSP or p-step-ahead predictor form of the system output $\hat{y}(t+1|t-p+1)$ given available system information until up to including the time instant $(t-p+1)$ is obtained from the MIMO ARX model of the system by rewriting the system model (equation 2.1) in the MSP prediction form. If the MSP is performed recursively, the form of the ARX model requires measurements at time instants $(t-p+2), \dots, t$. Assuming that $n_a < p$ implies that the prediction of the MIMO ARX model must be used in place of the measurements at the time instants $(t-p+2), \dots, t$ because the measurements are available only up to the time instant $(t-p+1)$. Thus, the multi-step-ahead MIMO ARX predictor is given by equation:

$$\begin{aligned} \hat{y}(t+1|t-p+1) = & \mathbf{A}_1 \hat{y}(t|t-p+1) + \dots + \mathbf{A}_{n_y} \hat{y}(t-n_y+1|t-p+1) \\ & + \mathbf{B}_1 \mathbf{u}(t) + \dots + \mathbf{B}_{n_u} \mathbf{u}(t-n_u+1), \end{aligned} \quad (2.6)$$

where $\mathbf{y}(t)$ and $\mathbf{u}(t)$, are the output and input vectors of the system n_y and n_u are the number of output and input lags respectively, \mathbf{A}_i for $i = 1, 2, \dots, n_y$ is the set of output coefficient vectors, \mathbf{B}_i for $i = 1, 2, \dots, n_u$ is the set of input coefficient vectors. Similarly, the multi-step-ahead SISO ARX predictor is expressed as:

$$\begin{aligned} \hat{y}(t+1|t-p+1) = & a_1 \hat{y}(t|t-p+1) + \dots + a_{n_y} \hat{y}(t-n_y+1|t-p+1) \\ & + b_1 u(t) + \dots + b_{n_u} u(t-n_u+1). \end{aligned} \quad (2.7)$$

The MIMO single-step-ahead and multi-step-ahead ARX predictors are shown in Figure 2.2.

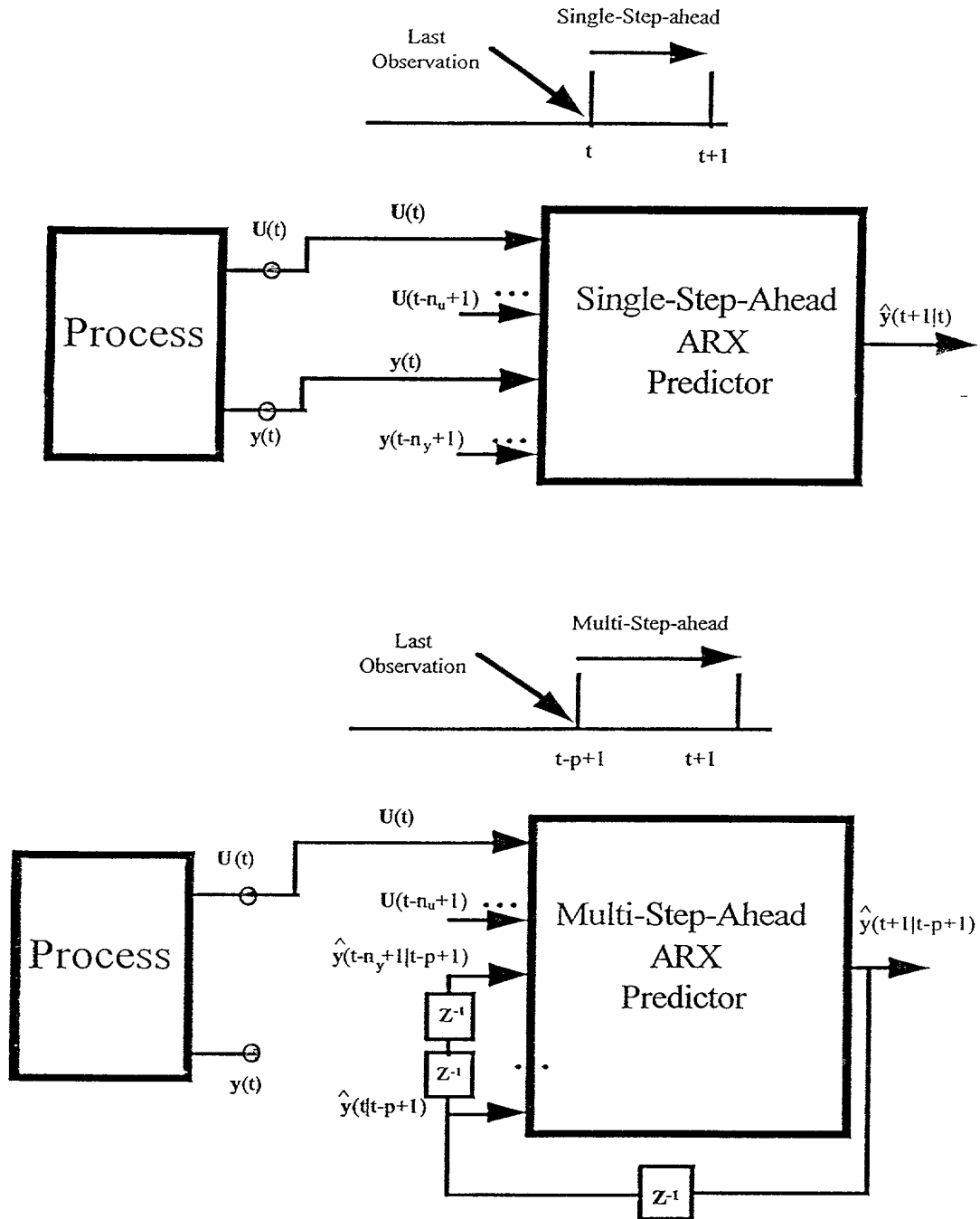


Figure 2.2: Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARX Predictors.

II.3.1.2 ARMAX System Models

II.3.1.2.1 Probabilistic Form of ARMAX

A more general linear system model is the Auto-Regressive Moving Average with eXogenous inputs (ARMAX). Compared to ARX system model given in equation (2.1), ARMAX system models include the disturbance terms which are the difference between the actual output and model predictions. Thus, ARMAX system models have more freedom to describe the properties of the disturbance term. The ARMAX model, for a MIMO system is represented as:

$$\begin{aligned} \mathbf{y}(t+1) = & \mathbf{A}_1 \mathbf{y}(t) + \cdots + \mathbf{A}_{n_y} \mathbf{y}(t-n_y+1) + \mathbf{B}_1 \mathbf{u}(t) + \cdots + \mathbf{B}_{n_u} \mathbf{u}(t-n_u+1) \\ & + \mathbf{e}(t) + \mathbf{C}_1 \mathbf{e}(t-1) + \cdots + \mathbf{C}_{n_e} \mathbf{e}(t-n_e+1) \end{aligned} \quad (2.8)$$

where

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix} \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad \mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ \vdots \\ e_n(t) \end{bmatrix} \quad (2.9)$$

n and m are the number of outputs and inputs respectively, $\mathbf{y}(t)$ is the output vector of the ARMAX system model, n_y is the number of past outputs (lag terms), $\mathbf{u}(t)$ is the input vector to the ARMAX system model, n_u is the number of past inputs (lag terms) in the model, and $\mathbf{e}(t)$ is the assumed noise vector with a zero mean and Gaussian distribution, n_e is the number of past noise terms (lag terms). The coefficients of the ARMAX model, A_i for $i = 1, 2, \dots, n_y$, B_i for $i = 1, 2, \dots, n_u$ and C_i for $i = 1, 2, \dots, n_e$ are assumed to be known.

The single-input single-output (SISO) ARMAX model is represented as:

$$y(t+1) = a_1 y(t) + \cdots + a_{n_y} y(t-n_y+1) + b_1 u(t) + \cdots + b_{n_u} u(t-n_u+1)$$

$$+ e(t) + c_1 e(t-1) + \cdots + c_{n_c} e(t-n_c+1), \quad (2.10)$$

where all the variables have been defined in equation (2.9).

II.3.1.1.2 Predictor Form of ARMAX

To use an ARMAX system model for forecasting the single-step-ahead prediction (SSP) or the multi-step-ahead prediction (MSP) form of the ARMAX system model must be derived.

The SSP form of the MIMO ARMAX system model given in equation (2.8) is:

$$\begin{aligned} \hat{\mathbf{y}}(t+1|t) = & \mathbf{A}_1 \mathbf{y}(t) + \cdots + \mathbf{A}_{n_y} \mathbf{y}(t-n_y+1) + \mathbf{B}_1 \mathbf{u}(t) + \cdots + \mathbf{B}_{n_u} \mathbf{u}(t-n_u+1) \\ & + \mathbf{C}_1 \boldsymbol{\varepsilon}(t-1) + \cdots + \mathbf{C}_{n_c} \boldsymbol{\varepsilon}(t-n_c+1) \end{aligned} \quad (2.11)$$

where $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are the output and input vectors, and $\boldsymbol{\varepsilon}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t|t-1)$ is the *prediction error* or the *residual* term vector. All other terms have been defined in equation (2.8).

Using the SISO ARMAX model structure given by equation (2.10), the corresponding SSP of the system output, $\hat{y}(t+1|t)$, is obtained as:

$$\begin{aligned} \hat{y}(t+1|t) = & a_1 y(t) + \cdots + a_{n_y} y(t-n_y+1) + b_1 u(t) + \cdots + b_{n_u} u(t-n_u+1) \\ & + c_1 u(t) + \cdots + c_{n_c} u(t-n_c+1). \end{aligned} \quad (2.12)$$

It can be shown that MIMO and SISO versions of MSP ARMAX predictor are of the same form as the MIMO and SISO versions of MSP ARX predictor given in equations (2.6) and (2.7) [45].

The MIMO single-step-ahead and multi-step-ahead predictors of ARMAX system are shown in Figure 2.3.

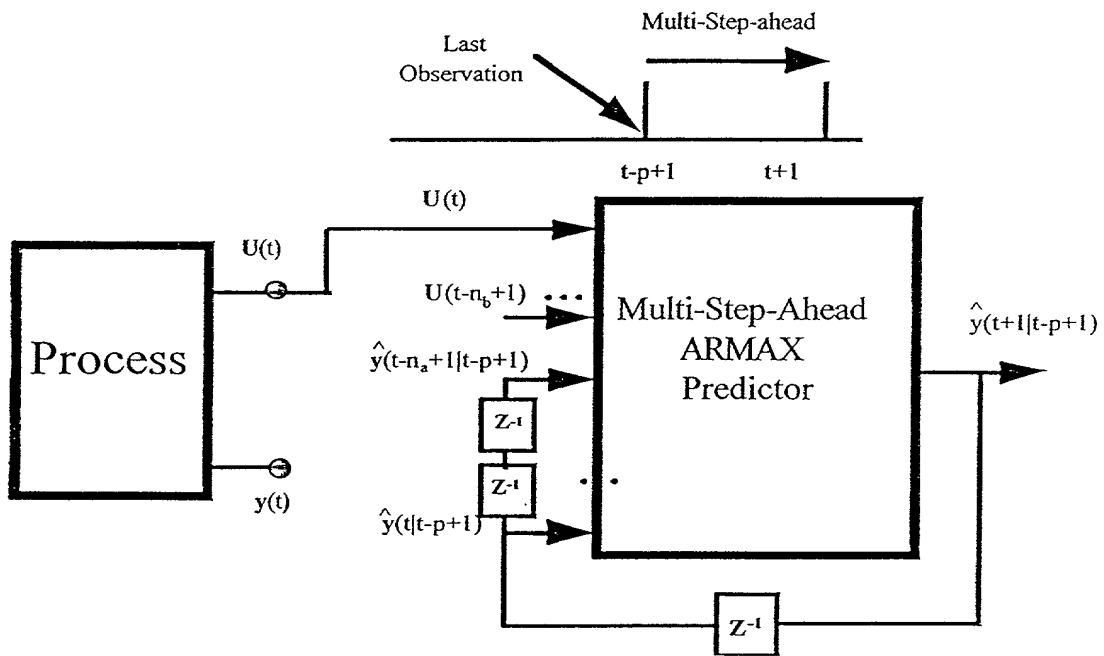
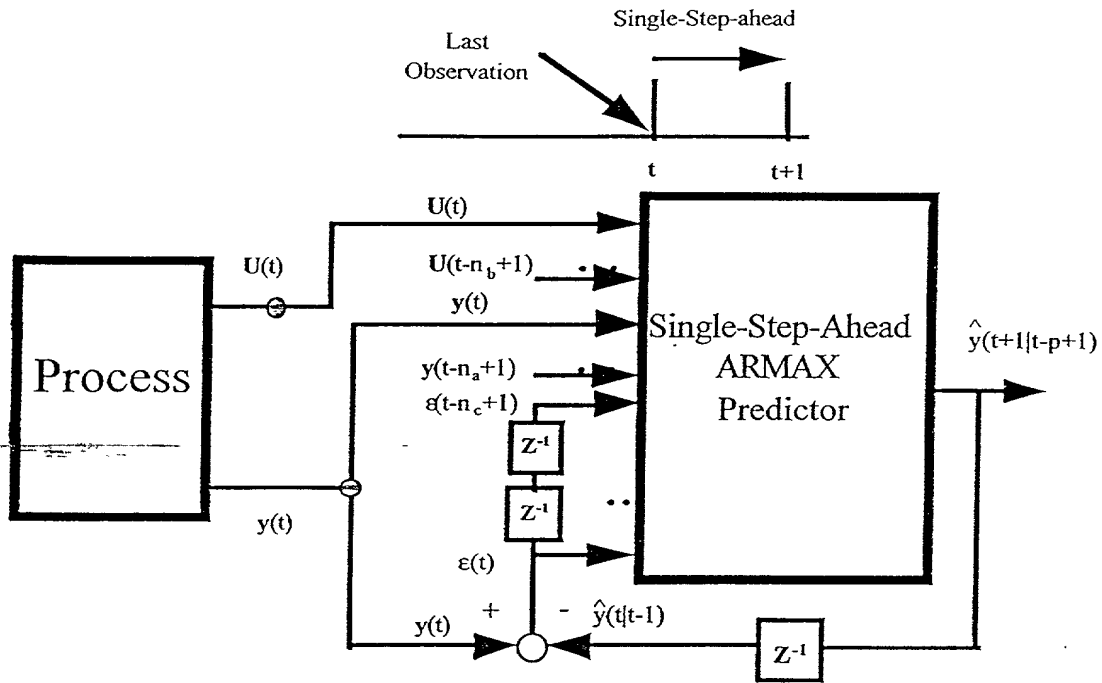


Figure 2.3: Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead ARMAX Predictors.

II.3.2 Nonlinear System Models

The majority of real world processes are nonlinear in nature. However, traditionally linear systems have been used to model such processes. The main reason behind such plans is the adequacy of linear systems to model nonlinear system, when the operation is in a limited linear. Moreover, the accurate identification of nonlinear system parameters is difficult. However, it is natural to attempt to model the nonlinear processes using nonlinear techniques. The advantage of nonlinear modeling techniques is their potentially greater accuracy in modeling and predicting nonlinear processes. The disadvantage of nonlinear system models is the difficulty in identifying correct model structures and estimating parameters. The adequacy of linear models in representing nonlinear processes depends on the nature of the problem. In general control problems, the prediction of the linear model is usually corrected with the signal received from the sensor in the next time step. Therefore inaccuracy in the prediction from the system model will not propagate through time and the system model need not be extremely accurate. However, in the forecasting problems the energy demand is predicted for several years into future. Therefore error in the forecast of a particular year will effect the forecast of the energy demand in the following years. Therefore, a linear system model may result in substantial errors.

The most common nonlinear system model is the Nonlinear Auto-Regressive with eXogenous input (NARX) [45], [18]. Conventional and neural networks (NN) expansions of the NARX have been reported in the literature [9]. However, neural

networks (NN) have shown to be more general expansions than the conventional expansions.

II.3.2.1 NARX System Models

II.3.2.1.1 NARX Probabilistic Forms

One of the most common nonlinear input-output models is the Nonlinear Auto-Regressive with eXogenous input (NARX). The NARX model of a nonlinear system has the following form:

$$\mathbf{y}(t+1) = \mathbf{f}(\mathbf{u}(t)) + \mathbf{e}(t+1) \quad (2.13)$$

where

$$\mathbf{u}(t) = \left[y(t), \dots, y(t-n_y+1), u(t), \dots, u(t-n_u+1) \right]^T, \quad (2.14)$$

and

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix}, \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}, \quad \mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ \vdots \\ e_n(t) \end{bmatrix}. \quad (2.15)$$

n and m are the number of outputs and inputs respectively, n_y and n_u are the number of past output and input lags, $\mathbf{e}(t)$ is the noise term vector with a zero mean and white Gaussian distribution, and $\mathbf{f}(\cdot)$ is a vector valued nonlinear function.

The NARX model can be written in terms of the individual inputs and output as [8]:

$$\begin{aligned} y_i(t+1) = & f_i(y_1(t), \dots, y_1(t-n_y^1+1), \dots, y_n(t), \dots, y_n(t-n_y^n+1), \\ & u_1(t), \dots, u_1(t-n_u^1+1), \dots, u_m(t), \dots, u_m(t-n_u^m+1)) \\ & + e_i(t+1), \quad \text{for } i=1, \dots, n \end{aligned} \quad (2.16)$$

If the NARX system is a SISO system, the system model will become:

$$\begin{aligned} y(t+1) = & f[y(t), y(t-1), \dots, y(t-n_y), \\ & u(t), u(t-1), \dots, u(t-n_u)] + e(t+1) \end{aligned} \quad (2.17)$$

where all the variables have been defined in equations (2.14) and (2.15).

II.3.2.1.2 NARX Predictor Forms

The general SSP form of the NARX system model given by equation (2.16) can be written as:

$$\begin{aligned} \hat{y}_i(t+1|t) = & f_i(y_1(t), \dots, y_1(t-n_y^1+1), \dots, y_n(t), \dots, y_n(t-n_y^n+1), \\ & u_1(t), \dots, u_1(t-n_u^1+1), \dots, u_m(t), \dots, u_m(t-n_u^m+1)) \end{aligned} \quad (2.18)$$

for $i=1, \dots, n$. For an equal number of delayed inputs and outputs for each of n outputs approximated, that is if $n_y^1 = n_y^2 = \dots = n_y^n = n_y$, and $n_u^1 = n_u^2 = \dots = n_u^m = n_u$, the following compact NARX SSP form is obtained:

$$\hat{y}_i(t+1|t) = f_i(\mathbf{u}(t)) \quad (2.19)$$

for $i=1, \dots, n$, where $\mathbf{u}(t)$ was defined in equation (2.14).

The MSP or p-step-ahead prediction of the output $\hat{y}(t+1|t-p+1)$ given the measurement data up to and including the time instant $(t-p+1)$ is obtained from the NARX model given by equation (2.19). However, since the NARX predictor requires measurements at the time instants $(t-p+1), \dots, t$, and these measurements are not available, the predictions of the NARX model must be used instead. The MSP form of the NARX model is expressed as:

$$\hat{y}_i(t+1|t-p+1) = f_i(\hat{\mathbf{u}}(t)), \quad \text{for } i = 1, \dots, n \quad (2.20)$$

where

$$\hat{\mathbf{u}}(t) = [\hat{y}(t|t-p+1), \dots, \hat{y}(t-n_y+1|t-p+1), \mathbf{u}(t), \dots, \mathbf{u}(t-n_u+1)]^T \quad (2.21)$$

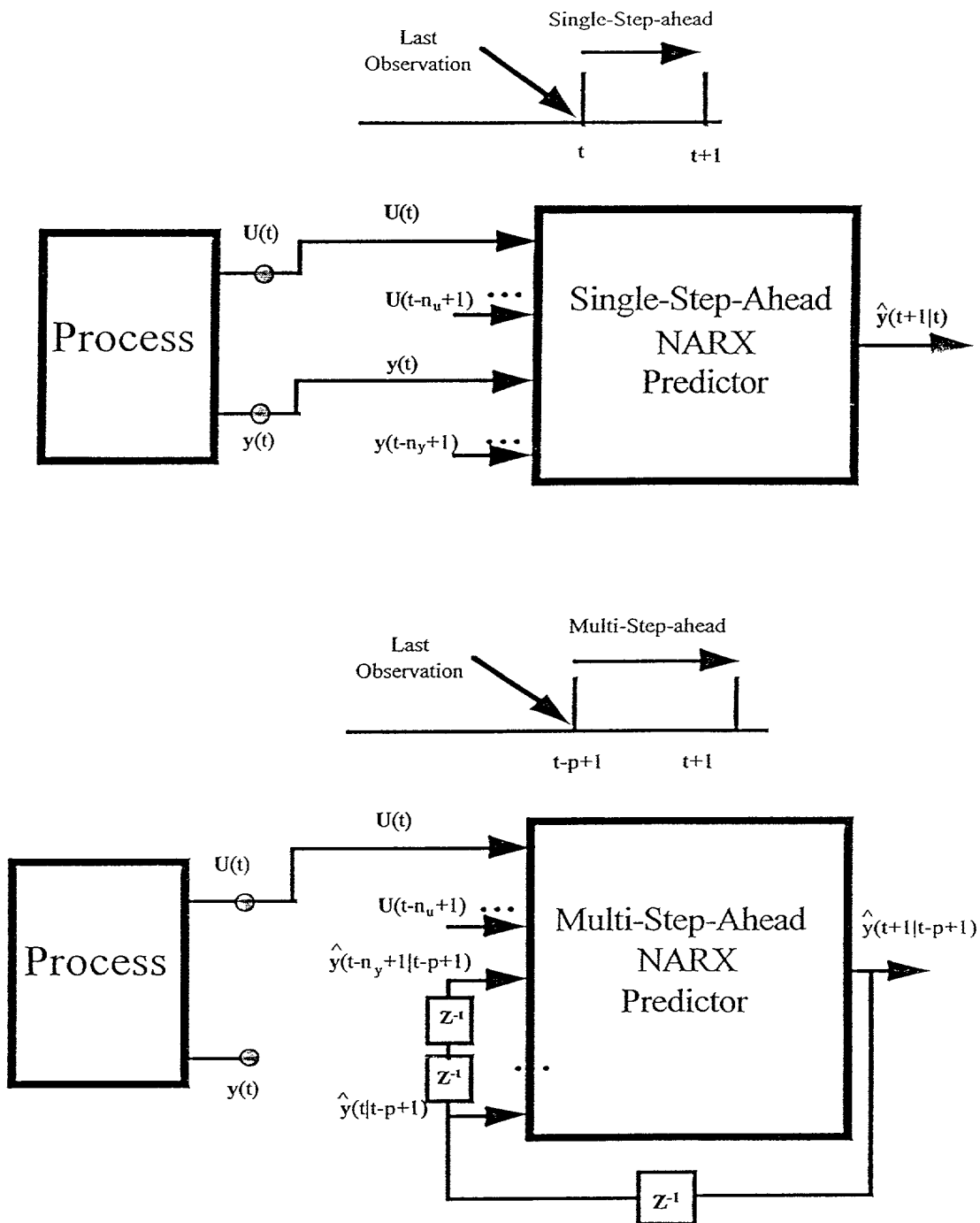


Figure 2.4: Block Diagrams of Single-Step-Ahead and Multi-Step-Ahead NARX Predictors.

and it is assumed that $n_y < p$. Block diagram of SSP and MSP predictors of NARX system models are shown in Figure 2.4

Two forms of nonlinear expansions of NARX are polynomial expansions and Neural Network (NN) expansions.

II.3.2.1.3 Polynomial Expansions of NARX

Recall that Taylor series expansion of a nonlinear system is:

$$f(x) \approx \sum_{i=0}^n \alpha_i x^i \quad (2.22)$$

where x denotes the independent variable. The most common polynomial expansions for nonlinearities account for the first linear component of the Taylor series expansions. Based on this, the conventional approximate expansions of nonlinearities given in equations (2.20) and (2.21) can be written as in equation (2.22).

The general form of the NARX predictor is given in equation (2.19) where the nonlinear function $f(\cdot)$ is unknown and must be determined. Billings [18] proposed the following polynomial parametrization of the NARX predictor:

$$\begin{aligned} \hat{y}_{poly,i}(t+1|t) = & \theta_0^{(i)} + \sum_{j_1=1}^q \sum_{j_2=j_1}^q \theta_{j_1 j_2}^{(i)} x_{j_1}(t) x_{j_2}(t) + \dots \\ & + \sum_{j_1=1}^q \dots \sum_{j_p=j_{p-1}}^q \theta_{j_1 \dots j_p}^{(i)} x_{j_1}(t) \dots x_{j_p}(t) \end{aligned} \quad (2.23)$$

for $i = 1, \dots, n$ outputs of the NARX system model, where p is the degree of the polynomial expansion and

$$q = n \times n_y + m \times n_u \quad (2.24)$$

Assuming that an equal number of delayed input and output vector components is used in the expansion, then the parametric structure in equation (2.23) is called the *polynomial* NARX.

II.3.2.1.4 Neural Networks Expansions of NARX

It has been shown that Neural Networks (NN) are promising tools in developing predictors, especially when multi-step-prediction (MSP) performance is of the primary interest. [16], [59]. In the NARX structures studied earlier, the number of model parameters is a direct function of model regressors. However, in NN the number of parameters is not a direct function of the number of regressors. Therefore, NN are generally categorized as semi-parametric model structures.

An Artificial Neural Network (ANN) is a biologically inspired topological network with multiple interconnected processing elements. These networks were initially introduced in the 1950's by McClelland and Pitts [56]. However, due to the unavailability of any systematic technique to identify their parameters these models did not find much support in scientific communities. It took approximately three decades until David Rumelhart [61] introduced a technique to identify the parameters of neural network models, for these models to become practical and popular. This estimation technique is known as the *backpropagation* technique.

Neural networks can be considered as input-output model structures, from an empirical modeling stand-point. The network is composed of at least 3 layers of neural processing elements. Each processing element is referred to as a node. Each layer consists of one or more nodes. The first and the last layers are referred to as the *input* and *output* layers respectively. The input and output layers are connected to each other through one or more layers of nodes which are referred to as hidden layers.

There are several known families of neural network structures. In the Feedforward Multilayer Perceptron (FMLP) family of networks, there is no connection

There are several known families of neural network structures. In the Feedforward Multilayer Perceptron (FMLP) family of networks, there is no connection

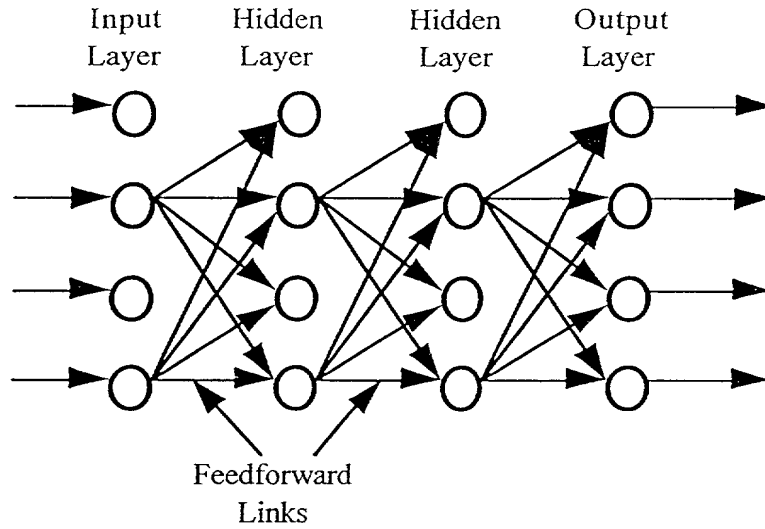


Figure 2.5: An FMLP Neural Network Architecture.

among the nodes within a layer and signals pass directly from one layer to the next layer. In Recurrent Multilayer Perceptron (RMLP) family of networks, there exists connections among the nodes within a layer. A hidden node receives signal from other nodes within the same layer in addition to nodes from the previous layer [59]. The NN's studied in this study are exclusively Feedforward Multilayer Perceptrons (FMLP). Figure 2.5 shows a FMLP NN.

The strength of the connections between nodes is referred to as a *weight*. In addition to connections from the nodes in the previous layer each node in a layer receives input from a *bias* node. In a fully connected network, the input layer simply fans out the input signals to all the nodes in the first hidden layer. Generally, the input and the output layer nodes have linear activation functions, while the hidden layer nodes have sigmoid-type activation functions.

In an FMLP, the output of each node is governed by the following function:

$$x_{[\ell,i]}(t) = \sigma_{[\ell,i]} \left(\sum_{j=1}^{N_{[\ell-1]}} \omega_{[\ell-1,j][\ell,i]} x_{[\ell-1,j]}(t) + b_{[\ell,i]} \right), \quad (2.25)$$

for nodes $i = 1, \dots, N_{[\ell]}$ in layer N , and $\ell = 1, \dots, L$ are the layers of the network. $\omega_{[\ell-1,j][\ell,i]}$ is the connection weight connecting j th node of layer $\ell - 1$ to i th node in layer ℓ . $b_{[\ell,i]}$ is the bias term connected to i th node of layer ℓ . $\sigma_{[\ell,i]}$ is the activation function of i th node in layer ℓ . Generally, all the nodes in a layer have the same form of activation function.

The unknown parameters in a FMLP network are the values of the weights and biases. The process of estimating these parameters is known as training. Through the training process, based on an appropriate learning algorithm, the optimal values of the weights and biases for a selected objective function are obtained [61], [50].

The general form of a NARX single-step-predictor (SSP) was given in equation (2.19) as:

$$\hat{y}(t + 1|t) = f(\mathbf{u}(t)) \quad (2.26)$$

where $\mathbf{u}(t)$, $\hat{y}(\cdot)$, $f(\cdot)$ have been defined earlier. To write a NN as a SSP, the nonlinear function $f(\cdot)$ can be written as:

$$f(\mathbf{u}(t)) \approx \sum_{i=1}^N \omega_i \sigma_i(\mathbf{u}(t)) + b \quad (2.27)$$

where ω_i is the i th weight, σ_i is the i th layer activation function and $\mathbf{u}(t)$ is the input vector. Based on the above, we can write the SSP with the NN approximation as:

$$\hat{y}(t + 1|t) = \sum_{i=1}^N \omega_i \sigma_i(\mathbf{u}(t)) + b \quad (2.28)$$

where N is the number of nodes in the hidden layer. A graphical representation of equation (2.28) is given in Fig. 2.6. The NN SSP predictor presented by equation (2.28)

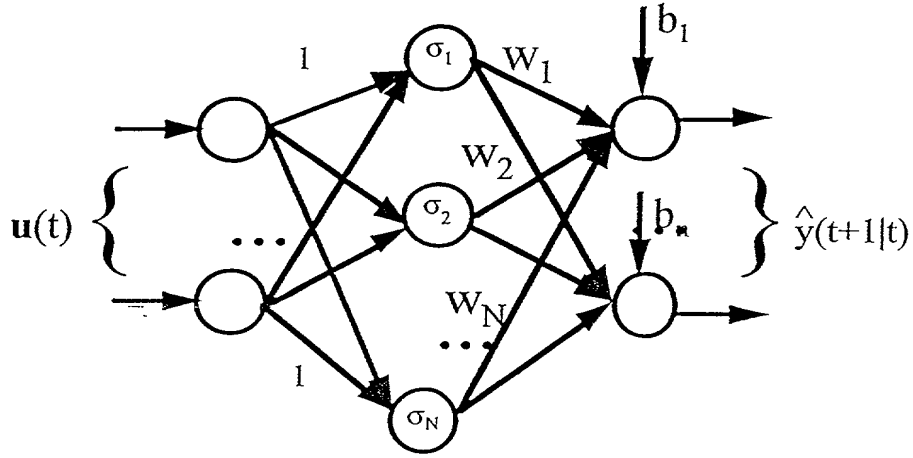


Figure 2.6: A Neural Network Predictor Representation.

can be decomposed for inputs and outputs of different layers. The input to FMLP input layer can be set to:

$$x_{[1]}(t+1) = u(t) \quad (2.29)$$

then the equations for the hidden layer and output layer of the FMLP depicted in Figure 2.6 can be written as:

$$x_{[2,j]}(t+1) = \sigma_{[2,j]} \left(\sum_{k=1}^{N_{[1]}} \omega_{[1,k][2,j]} x_{[1,k]}(t+1) + b_{[2,j]} \right) \quad (2.30)$$

$$x_{[3,i]}(t+1) = \sigma_{[3,i]} \left(\sum_{j=1}^{N_{[2]}} \omega_{[2,j][3,i]} x_{[2,j]}(t+1) + b_{[3,i]} \right) \quad (2.31)$$

for $j = 1, \dots, N_{[2]}$, and $i = 1, \dots, N_{[3]}$. The output of the NN can be expressed as the SSP predictor:

$$\hat{y}_{NN}(t + 1|t) = x_{[3]}(t + 1) \quad (2.32)$$

or in compact form, above set of equations can be written as:

$$\hat{y}_{NN}(t + 1|t) = F(u(t)) \quad (2.33)$$

and

$$F(u(t)) = w_{2 \rightarrow 3} \sigma_{[2]}(w_{1 \rightarrow 2} u(t) + b_{[2]}) + b_{[3]} \quad (2.34)$$

where $w_{1 \rightarrow 2}$ and $w_{2 \rightarrow 3}$ are the weight matrices connecting layers $1 \rightarrow 2$ and $2 \rightarrow 3$ respectively, and $b_{[2]}$, $b_{[3]}$ are the bias vectors to layer 2 and layer 3 respectively.

For a NN with a general L number of layers, with $N_{[i]}$ nodes in each layer, the following functions can be defined:

$$F_{[i+1,j]}(x_{[i]}(t+1)) = \sum_{k=1}^{N_{[i]}} \omega_{[i,k]} \sigma_{[i,k]}(x_{[i,k]}(t+1)) + b_{[i+1,j]} \quad (2.35)$$

$$F_i = \left[F_{[i,1]}, F_{[i,2]}, \dots, F_{[i,N_{[i+1]}]} \right]^T \quad i = 1, \dots, (L-1) \quad (2.36)$$

where

$$\sigma_{[i,i]} = \begin{cases} x & \text{for } i = 1 \\ \tanh(x) & \text{for all other } i \end{cases} \quad (2.37)$$

then the general form of SSP for a FMLP with L layers can be written as:

$$\hat{y}_{NN}(t + 1|t) = F_{(L-1)}(F_{(L-2)}, \dots, F_i(F_{(i-1)}, \dots, F_1(u(t)))) \quad (2.38)$$

where all the variables have been defined earlier.

It is known that the success of NNs in general prediction problems is due to the nonlinear approximations afforded by NN predictors such as sigmoid and hyperbolic tangent forms. It has been reported that a NN with a single hidden layer can approximate any nonlinear function given a sufficient number of hidden nodes [3].

The approach developed for SSP can be extrapolated to a NN predictor for MSP in the form:

$$\hat{y}_{NN}(t+1|t-p+1) = F(\hat{u}(t)) \quad (2.39)$$

Equations (2.38) and (2.39) can be used recursively to develop a MSP. It should be mentioned that in developing a SSP or MSP using NN, the number of layers L , the number of nodes in each layer $N_{[l]}$, and the value of the elements of the weight vector, *i.e.* $\omega_{[j][l+1,k]}$ must be determined. This is the subject of System Identification (SI) which will be discussed in the next section. Figure 2.7 shows schematic diagrams of SSP and MSP NN predictors.

II.4 System Identification

System Identification (SI) refers to systematic procedures for development of mathematical models of a process or system. While tools such as linear and nonlinear regression have been around for years, only recently have all these techniques been combined to produce a systematic procedure to generate robust models of physical systems.

Several approaches exist for developing a model for a system [45]. The *prediction error methods* minimize the size of the error between the system output and the model output. The *correlation methods* minimize the correlation between the model residuals

and each exogenous variable. Existence of a correlation in residuals indicates that the derived model has not extracted all the information from the data set.

In the following subsections the *prediction error* method is applied to system models described in previous section. Linear and nonlinear SI techniques are described. Least squares estimation techniques, maximum likelihood estimation and instrument variables will be described and applied to linear systems.

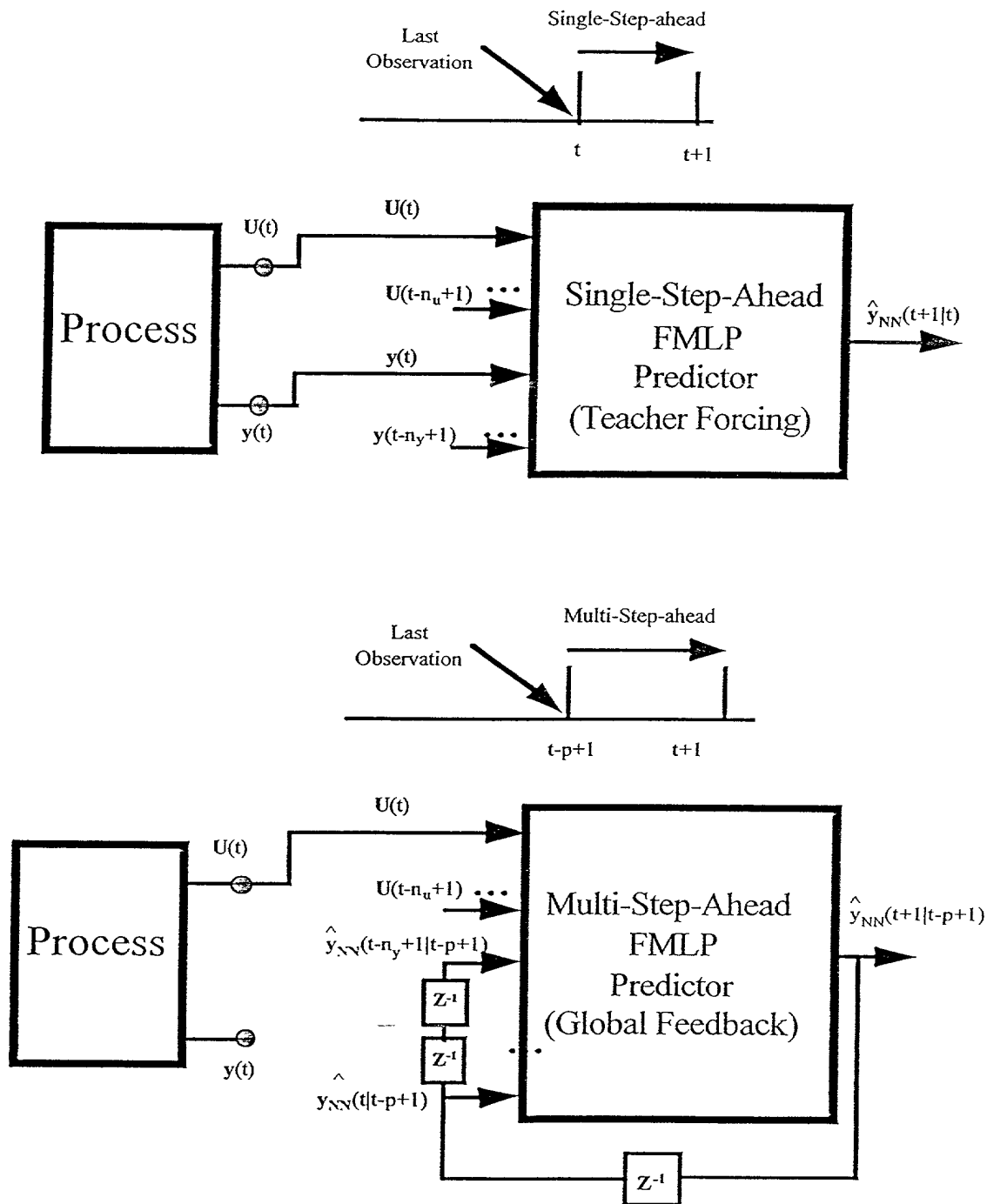


Figure 2.7: Block Diagrams of the Single-Step-Ahead and Multi-Step-Ahead FMLP Neural Networks (NN) Predictors.

Nonlinear least-squares estimation and backpropagation techniques will be addressed and applied to nonlinear systems.

II.4.1 Linear System Identification

Linear SI techniques are procedures used to identify model structures of linear systems. There is a vast amount of literature on linear system identification techniques [45], [55]. These techniques can be applied to MIMO and SISO systems.

The prediction error of a certain process model is described as:

$$\varepsilon(t, \hat{\theta}) = y(t) - \hat{y}(t / \hat{\theta}) \quad (2.40)$$

where $y(t)$ is the process output and $\hat{y}(t / \hat{\theta})$ is the model output for a particular set of parameter estimates $\hat{\theta}$. Then, in general, the estimation problem objective function is defined as:

$$V(\theta, Z^N) = \frac{1}{N} \sum_{i=1}^N f(\varepsilon_i(t, \theta)) \quad (2.41)$$

where $f(\cdot)$ is a scalar positive valued function and Z^N is the information vector including the input and output observations.

The prediction-error-models try to minimize the prediction error by computing the optimal estimates of parameter θ of the model. The optimal values of θ will result in minimum value of the objective function V . Different estimation techniques have been developed, based on the selection of the $f(\cdot)$.

II.4.1.1 Least Squares Estimation (LSE)

Least Squares Estimation (LSE) replaces the $f(\cdot)$ in equation (2.41), with a squared term and the objective function to be minimized becomes:

$$V(\theta, Z^N) = \frac{1}{N} \sum_{i=1}^N (\varepsilon_i(t, \theta))^2 \quad (2.42)$$

The predictor form of an ARX model was given in (2.5). If the model parameters are denoted as θ , and the information vector as $\varphi(t)$, the predictor form of ARX model can be written as:

$$\hat{y}(t+1|t) = \varphi^T(t+1)\theta, \quad (2.43)$$

where

$$\varphi(t+1) = [y(t), \dots, y(t-n_y+1), u(t), \dots, u(t-n_u+1)]^T \quad (2.44)$$

$$\theta = [a_1, \dots, a_{n_y}, b_1, \dots, b_{n_u}]^T \quad (2.45)$$

This is a regression model representation of an ARX model with the regressor vector θ and information vector $\varphi(t+1)$. The objective function of the LSE formulation (equation 2.42) for the ARX model structure will therefore be:

$$V(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^N [y(t) - \hat{y}(t|t-1; \theta)]^2 \quad (2.46)$$

The parameters of the ARX model are obtained by minimizing the above objective function with respect to θ . The solution of the above LSE for the estimated parameters $\hat{\theta}_N$ is:

$$\hat{\theta}_N = \left[\sum_{t=1}^N \varphi(t) \varphi^T(t) \right]^{-1} \sum_{t=1}^N \varphi(t) y(t) \quad (2.47)$$

A similar procedure can be followed to compute the parameters of an ARMAX model predictor (equation 2.12). However, the information vector and the regressor vector will be defined as:

$$\varphi(t+1; \theta) = \begin{bmatrix} y(t), \dots, y(t-n_y+1), u(t), \dots, u(t-n_u+1), \\ \varepsilon(t; \theta), \dots, \varepsilon(t-n_y+1; \theta) \end{bmatrix}^T \quad (2.48)$$

$$\theta = \begin{bmatrix} a_1, \dots, a_{n_y}, b_1, \dots, b_{n_u}, c_1, \dots, c_{n_e} \end{bmatrix}^T \quad (2.49)$$

where

$$\varepsilon(t; \theta) = y(t) - \hat{y}(t-1; \theta) \quad (2.50)$$

In the least square estimation of ARMAX model structures, the information vector $\varphi(t+1)$ is dependent on the parameter vector θ through the prediction error term $\varepsilon(t; \theta)$.

II.4.1.2 Maximum Likelihood Estimation (MLE)

The Maximum Likelihood Estimator (MLE) can be expressed in a stochastic environment [45]. Let y^N be a random variable:

$$y^N = (y(1), y(2), \dots, y(n)) \quad (2.51)$$

Suppose the probability density function of y^N is:

$$f(\theta; x_1, x_2, \dots, x_N) = f_y(\theta; x^N) \quad (2.52)$$

That is:

$$P(y^N \in A) = \int_{d^N \in A} f_y(\theta; x^N) dx^N \quad (2.53)$$

The parameters of the estimator are determined using the available information set of y^N , that is y_*^N . Then, the MLE will be the parameters which will maximize the following description:

$$\hat{\theta}_{ML}(y_*^N) = \arg \max_{\theta} f_y(\theta; y_*^N) \quad (2.54)$$

II.4.1.3 Instrument Variables Method

The Instrument Variable (IV) method is similar to the LSE method. Suppose that the observed data have actually been generated by:

$$y(t) = \varphi^T(t; \theta) \theta_o + v_o(t) \quad (2.55)$$

for some sequence $v_o(t)$. In general, parameters $\hat{\theta}_N$ obtained using LSE are not close to θ . This is due to the correlation between $v_o(t)$ and $\varphi(t)$. A correlation vector $\zeta(t)$ can be added to the LSE scheme such that:

$$\hat{\theta}_N^{LS} = \text{Sol} \left\{ \frac{1}{N} \sum_{t=1}^N \varphi(t) [y(t) - \varphi^T(t) \theta] = 0 \right\} \quad (2.56)$$

The elements of $\zeta(t)$ are called instrument variables (IV). The IV should be correlated with the regression variables and uncorrelated with the noise vector $v_o(t)$.

$$E \{ \zeta(t), \varphi^T(t) \} > 0 \quad (2.57)$$

$$E \{ \zeta(t), v_o(t) \} = 0 \quad (2.58)$$

Thus, the IV solution will be:

$$\hat{\theta}_N^{IV} = \text{Sol} \left\{ \frac{1}{N} \sum_{t=1}^N \zeta(t) [y(t) - \varphi^T(t) \theta] = 0 \right\} \quad (2.59)$$

or

$$\hat{\theta}_N^{IV} = \left[\frac{1}{N} \sum_{t=1}^N \zeta(t) \varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^N \zeta(t) y(t) \quad (2.60)$$

II.4.2 Nonlinear System Identification

The majority of real world processes are nonlinear. In the past linear SI tools have been used under restricted conditions for the identification and treatment of nonlinear processes. However, with the advancement of theoretical foundations and the availability of fast computers, nonlinear SI techniques are now more successfully applied in the identification and treatment of nonlinear processes.

II.4.2.1 Nonlinear Least Squares Estimation

Recall the Billings [18] proposed polynomial parametrization of the NARX predictors given in equation (2.23):

$$\begin{aligned} \hat{y}_{poly,i}(t+1|t) = & \theta_o^{(i)} + \sum_{j1=1}^q \sum_{j2=j1}^q \theta_{j1j2}^{(i)} x_{j1}(t) x_{j2}(t) + \dots \\ & + \sum_{j1=1}^q \dots \sum_{j2=j_p-1}^q \theta_{j1 \dots j_p}^{(i)} x_{j1}(t) \dots x_{j_p}(t) \end{aligned} \quad (2.61)$$

where the variables have been defined in equation (2.23). This polynomial expansion can be written in regression form as:

$$z(t) = \sum_{i=1}^M p_i(t) \theta_i + \zeta(t), \quad (2.62)$$

for $t = 1, \dots, N$. N is the data length, $p_i(t)$ are polynomials of $x_1(t)$ to $x_n(t)$ up to degree ℓ , $\zeta(t)$ is the modeling error, θ_i are unknown parameters to be estimated, and M is the total number of terms in the polynomial. Billings [18] proposed that each of the n system outputs be independently approximated, using the expansion given in equation (2.60). Thus, the identification of the m -input n -output system would be converted to the identification of n , m -input single output systems.

The orthogonal least-squares estimation technique can be used in identification of this system, because of the following reasons:

1. Parameter estimation methods using least-squares are well developed and applicable to the polynomial NARX model structure.
2. Nonlinear systems can be expressed using a polynomial expansion in which the parameters enter linearly.

Equation (2.62) can be written in matrix form as:

$$\mathbf{Z} = \mathbf{P}\theta + \Xi \quad (2.63)$$

with

$$\mathbf{Z} = [z(1) \ z(2) \ \dots \ z(N)]^T, \quad (2.64)$$

$$\mathbf{P} = [p_1 \ p_2 \ \dots \ p_M], \quad (2.65)$$

$$\theta = [\theta_1 \ \theta_2 \ \dots \ \theta_M]^T, \quad (2.66)$$

$$\Xi = [\zeta(1) \ \zeta(1) \ \dots \ \zeta(N)]^T, \quad (2.67)$$

and

$$p_i = [p_i(1) \ p_i(2) \ \dots \ p_i(N)], \quad i = 1, \dots, M \quad (2.68)$$

A linear least-squares estimation algorithm can be applied to obtain the parameters $\hat{\theta}$ which minimize $\|\mathbf{Z} - \mathbf{P}\theta\|$, where $\|\cdot\|$ is the Euclidean norm. It is well known that the solution will satisfy the equation:

$$\mathbf{P}^T \mathbf{P} \theta = \mathbf{P}^T \mathbf{Z} \quad (2.69)$$

where $\mathbf{P}^T \mathbf{P}$ is called the information matrix. Further detail regarding the orthogonal least-squares estimation applied to NARX model structures can be found in [8].

II.4.2.2 Neural Networks Estimation

The backpropagation estimation algorithm was first introduced by Rumelhart to compute the parameters of neural networks (NN) model structures [61]. Having pairs of input-output observations, the neural networks (NN) predicts an output for each input sample presented to its input layer. The backpropagation learning algorithm uses the difference between the NN predicted output and the observed output to adjust the parameters of the NN model. The direction of the adjustment is based on the gradient descent algorithm. Repeating this process for all the available input and output pairs for a large number of iterations will adjust the NN outputs in good agreement with actual observed outputs.

Let a Feedforward Multilayer Perceptron (FMLP) network consist of L layers. Let each layer ℓ consist of $N_{[\ell]}$ nodes and the weights connecting node k in layer $\ell-1$ to node j in layer ℓ be denoted as $\omega_{[\ell-1,k][\ell,j]}$. Moreover, assume that the output layer nodes have linear discriminatory function, while the hidden layer nodes have sigmoid type discriminatory functions as given in equation (2.37). Let input and output include P patterns or sample pairs. The input and output pairs of observations can be denoted as:

$$\begin{aligned} Z^P &= [\mathbf{u}^P, \mathbf{y}^P] \\ \mathbf{u}^P &= [\underline{u}_1, \underline{u}_2, \dots, \underline{u}_P] \\ \mathbf{y}^P &= [\underline{y}_1, \underline{y}_2, \dots, \underline{y}_P] \end{aligned} \tag{2.70}$$

Each input pattern \underline{u} includes F_I features, and each output pattern includes F_O features. In the basic form of NN, the number of nodes in the first layer is equal to the

number of input features F_I and output nodes equal the number of output features F_O . Therefore, the number of nodes in the input and the output layer will be:

$$N_{[1]} = F_I \quad , \quad N_{[L]} = F_O \quad (2.71)$$

Generally, the NN weights are assigned initial values in the range -1 and 1 by a random process. Presenting the pattern p to the first layer, then:

$$x_{[1,i]} = \underline{u}_p(i) \quad \text{for } i = 1, \dots, N_{[1]} \quad (2.72)$$

The input layer fans out the input pattern among the nodes in the hidden layer. At each hidden node the weighted sum of input and the bias term is initially computed.

$$net_{[2,j]} = \sum_{i=1}^{N_{[1]}} \omega_{[1,i][2,j]} x_{[1,i]}^{(t+1)} + b_{[2,j]} \quad (2.73)$$

The output of each node is a sigmoid-type activation function form of the above weighted sum:

$$x_{[2,j]} = \frac{1}{1 + e^{-net_{[2,j]}}} \quad \text{for } j = 1, \dots, N_{[2]} \quad (2.74)$$

Assuming linear nodes in the output layer, the output for each input pattern \underline{u}_p is given by:

$$\hat{y}_{[3,k]} = x_{[3,k]} = \sum_{j=1}^{N_{[2]}} \omega_{[2,j][3,k]} x_{[2,j]} + b_{[3,k]} \quad (2.75)$$

The estimation process (training) now tries to minimize the difference between observed and computed outputs. Assuming that the output pattern includes only one feature, the objective function used in training the NN is of the form:

$$V(Z, \omega) = \frac{1}{2} \sum_{p=1}^P \sum_{k=1}^{N_{out}} (y_{[p,k]} - \hat{y}_{[p,k]})^2 \quad (2.76)$$

where the NN prediction for node k is given by equation (2.75).

The backpropagation training process minimizes the above objective function $V(Z, \omega)$ by adjusting the weights ω , using the gradient descent optimization method. The adjustment in the value of the individual weights is computed by differentiating the objective function with respect to each weight. In another words, the error in the output is propagated backward through the network and each weight is modified proportional to its contribution to the output. Mathematically, this is equivalent to the gradient of the objective function (equation 2.75) with respect to each weight.

In a 3 layer network, two set of weights vectors $W_{1 \rightarrow 2}$ and $W_{2 \rightarrow 3}$ should be accounted for. Differentiating equation (2.76) with respect to vector $W_{2 \rightarrow 3}$:

$$\frac{\partial V(Z, \omega)}{\partial \omega_{2 \rightarrow 3}} = (y_{[p,k]} - \hat{y}_{[p,k]}) \frac{\partial \hat{y}_{[p,k]}}{\partial \omega_{2 \rightarrow 3}} \quad (2.77)$$

Defining for each outer layer nodes:

$$\delta_k = (y_{[p,k]} - \hat{y}_{[p,k]}) \quad (2.78)$$

and from equation (2.75):

$$\frac{\partial \hat{y}_{[p,k]}}{\partial \omega_{2,j \rightarrow 3,k}} = x_{[2,j]} \quad (2.79)$$

Therefore, the gradient of the objective function with respect to weights in links to outer layer nodes is:

$$\frac{\partial V(Z, \omega)}{\partial \omega_{[2,j][3,k]}} = -\delta_k x_{[2,j]} \quad (2.80)$$

The adjustment in the weight $\omega_{[2,j][3,k]}$ is proportional to the negative of the above gradient value. The coefficient of the equality is referred to as the *learning rate* η . Therefore:

$$\Delta \omega_{[2,j][3,k]} = \eta \delta_k x_{[2,j]} \quad (2.81)$$

where η is a constant referred to as the learning rate. Compute the gradient with respect to $\mathbf{W}_{1 \rightarrow 2}$ requires considering error propagation through the hidden nodes of the second layer.

$$\frac{\partial V(Z, \omega)}{\partial \omega_{[1,i][2,j]}} = \frac{\partial V(Z, \omega)}{\partial x_{[2,j]}} \frac{\partial x_{[2,j]}}{\partial \omega_{[1,i][2,j]}} \quad (2.82)$$

$$\frac{\partial V(Z, \omega)}{\partial x_{[2,j]}} = - \sum_{k=1}^{N^{[3]}} \delta_k \omega_{[2,j][3,k]} \quad (2.83)$$

$$\frac{\partial x_{[2,j]}}{\partial \omega_{[1,i][2,j]}} = \frac{\partial x_{[2,j]}}{\partial \text{net}_{[2,j]}} \frac{\partial \text{net}_{[2,j]}}{\partial \omega_{[1,i][2,j]}} \quad (2.84)$$

Choosing the activation function as (2.74), then :

$$\frac{\partial x_{[2,j]}}{\partial \text{net}_{[2,j]}} = x_{[2,j]} (1 - x_{[2,j]}) \quad (2.85)$$

$$\frac{\partial \text{net}_{[2,j]}}{\partial \omega_{[1,i][2,j]}} = x_{[1,i]} \quad (2.86)$$

Combining the results of the above equations, the gradient of the objective function with respect to vector $\mathbf{W}_{1 \rightarrow 2}$ will be:

$$\frac{\partial V(Z, \omega)}{\partial \omega_{[1,j][2,j]}} = -\delta_{[2,j]} x_{[2,j]} (1 - x_{[2,j]}) x_{[1,i]} \quad (2.87)$$

and the change in the weights will be:

$$\Delta \omega_{[1,j][2,j]} = \eta \delta_{[2,j]} x_{[2,j]} (1 - x_{[2,j]}) x_{[1,i]} \quad (2.88)$$

Equations (2.81) and (2.88) gives the update equations for $W_{1 \rightarrow 2}$ and $W_{2 \rightarrow 3}$ vectors.

An additional term which is usually added to the weight update equation is the momentum term. The momentum term adds a positive change to the direction of minimization if two consecutive computations of the gradient have the same sign, and adds a negative change if the two gradient computations have opposite signs. The final weight update equation using the backpropagation learning algorithm is:

$$\Delta \omega = -\eta \frac{\partial V}{\partial \omega} + \alpha (\omega_{new} - \omega_{old}) \quad (2.89)$$

where

η	learning rate
α	momentum coefficient
ω_{new}	value of weights at previous iteration
ω_{old}	value of weights at 2nd previous iteration

The above procedure is repeated for many iterations until the weight changes become negligible.

Similar to any numerical minimization method, the backpropagation learning algorithm can fall into local minima, or diverge. In practice, the faster versions of backpropagation are usually used [50]. In many occasions the model developer stops the learning process before reaching a global minimum. In the next chapter, several forms of objective functions and stopping criteria will be discussed.

II.5 Regression Analysis

Although regression models can be classified as a form ARX and NARX model structures, but the power system community has known regression models for a long time. Like many other fundamental mathematical concepts, the idea of regression analysis seems to have been due in the first place to Gauss who outlined the method known as “Ordinary Least Square” (OLS). Through the centuries regression analysis has found a foot place in every engineering field.

The basic principle of regression analysis is establishing a cause-and-effect relationship between two sets of variables. The first set of variables, called exogenous (or independent or regressor) variables cause changes in the second set of variables, called endogenous (dependent) variable [55]. Such relationships can be found in every branch of science and engineering. However, the form of the relationship can be quite different depending on the application.

Regression analysis can be a two variable model (one independent and one dependent variable) or multi-variable (several independent variables and one dependent variable). Generally, it is assumed that different regressors are independent. Time series analysis is a form of regression analysis where a dependent variable is related to its previous values. Regression analysis can be linear or nonlinear.

II.5.1 Linear Regression Analysis

In linear regression analysis a linear relationship between the independent variables and the dependent variable is assumed.

Considering N observations of $k+1$ variables, a dependent variable and k independent variables (including the constant term) a linear regression model can be written as a series of N equations:

$$\begin{aligned} y_1 &= \beta_1 + \beta_2 x_{21} + \beta_3 x_{31} + \dots + \beta_k x_{k1} + \varepsilon_1 \\ y_2 &= \beta_1 + \beta_2 x_{22} + \beta_3 x_{32} + \dots + \beta_k x_{k2} + \varepsilon_2 \\ &\vdots \\ y_N &= \beta_1 + \beta_2 x_{2N} + \beta_3 x_{3N} + \dots + \beta_k x_{kN} + \varepsilon_N \end{aligned} \quad (2.90)$$

The corresponding matrix formulation of the model is:

$$\mathbf{Y} = \mathbf{X} \boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad (2.91)$$

where

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{21} & x_{31} & \dots & x_{k1} \\ 1 & x_{22} & x_{32} & \dots & x_{k2} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{2N} & x_{3N} & \dots & x_{kN} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_N \end{bmatrix} \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_N \end{bmatrix} \quad (2.92)$$

$\mathbf{Y} = N \times 1$ column vector of dependent variable observations,

$\mathbf{X} = N \times k$ matrix of independent variable observations,

$\beta = k \times 1$ column vector of unknown parameters,

\mathcal{E} = $N \times 1$ column vector of errors.

The assumptions of the classical linear regression model are:

1. The elements of \mathbf{X} are fixed and have finite variance. In addition \mathbf{X} has rank k , which is less than the number of observations N .

2. ε is normally distributed with $E\{\varepsilon\}=0$, and $E\{\varepsilon^T \varepsilon\}=\sigma^2 \mathbf{I}$, where \mathbf{I} is an $N \times N$ identity matrix.

The assumption that \mathbf{X} has rank k guarantees that perfect collinearity will not be present. With perfect collinearity one of columns of \mathbf{X} would be a linear combination of the remaining columns and the rank of \mathbf{X} would be less than k . In addition to the normality assumption of error terms, we assume that each error term has mean 0, and all variances are constant, and all covariances are 0.

II.5.1.1 Least Squares Estimation in Linear Regression Analysis

Our objective is to find a vector of parameters $\hat{\beta}$ which minimizes:

$$\text{ESS} = \sum_{i=1}^N \hat{\varepsilon}_i^2 = \hat{\varepsilon}^T \hat{\varepsilon} \quad (2.93)$$

where

$$\hat{\varepsilon} = \mathbf{Y} - \hat{\mathbf{Y}} \quad (2.94)$$

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta} \quad (2.95)$$

$\hat{\varepsilon}$ represent the $N \times 1$ vector of regression residuals, while $\hat{\mathbf{Y}}$ represents the $N \times 1$ vector of fitted values for \mathbf{Y} . Substituting from equations (2.94) and (2.95) into equation (2.93):

$$\hat{\varepsilon}^T \hat{\varepsilon} = (\mathbf{Y} - \mathbf{X}\hat{\beta})^T (\mathbf{Y} - \mathbf{X}\hat{\beta}) = \mathbf{Y}^T \mathbf{Y} - 2\hat{\beta}^T \mathbf{X}^T \mathbf{Y} + \hat{\beta}^T \mathbf{X}^T \mathbf{X} \hat{\beta} \quad (2.96)$$

The least-squares estimators are computed by differentiating ESS as:

$$\frac{\partial \text{ESS}}{\partial \hat{\beta}} = -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X} \hat{\beta} = 0 \quad (2.97)$$

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{Y}) \quad (2.98)$$

The matrix $\mathbf{X}'\mathbf{X}$ is called cross-product matrix, and is guaranteed to have an inverse because of the assumption that \mathbf{X} has rank k .

Among the properties of the linear least-squares estimator $\hat{\beta}$ the following can be briefly mentioned:

1. $\hat{\beta}$ is an unbiased estimator of β
2. $\hat{\beta}$ is a linear function of the actual observations of \mathbf{Y} .
3. $\hat{\beta}$ is the best linear unbiased estimator of β .

The variation of the original \mathbf{Y} observations has two components, one represents the explained portion by the regression model and captured the linear regression model, and the second, the unexplained and represented by the residual term. These two components are given in the right hand side of equation (2.99) respectively:

$$\mathbf{Y} = \mathbf{X}\hat{\beta} + \hat{\mathcal{E}} \quad (2.99)$$

The goodness of fit R^2 is the portion of the variations in \mathbf{Y} which is explained by the regression model [55]:

$$R^2 = 1 - \frac{\hat{\mathcal{E}}^T \mathcal{E}}{\mathbf{Y}^T \mathbf{Y}} = \frac{\hat{\beta}^T \mathbf{X}^T \mathbf{X} \hat{\beta}}{\mathbf{Y}^T \mathbf{Y}} \quad (2.100)$$

However, if the dependent variable does not have 0 mean, the definition of R^2 will be modified to:

$$R^2 = \frac{\hat{\beta}^T \mathbf{X}^T \mathbf{X} \hat{\beta} - N\bar{Y}^2}{\mathbf{y}^T \mathbf{y}} \quad (2.101)$$

where \bar{Y} is the mean of \mathbf{Y} , and :

$$y^T y = Y^T Y - N\bar{Y}^2 \quad (2.102)$$

II.5.2 Nonlinear Regression Analysis

While procedures for nonlinear estimation can be computationally expensive, Nonlinear regression models greatly increase the scope of model structure that can be fitted to data at the expense of more computationally expensive parameter estimation procedure [55], [4].

Consider the following nonlinear regression equation where f is a nonlinear function of k independent variables X_1, X_2, \dots, X_k and p coefficients $\beta_1, \beta_2, \dots, \beta_p$:

$$Y = f(X_1, X_2, \dots, X_k, \beta_1, \beta_2, \dots, \beta_p) + \varepsilon \quad (2.103)$$

Our concern is with equations that are *inherently* nonlinear. For example:

$$Y = \alpha_0 + \alpha_1 X_1^{\beta_1} + \alpha_2 X_2^{\beta_2} + \varepsilon \quad (2.104)$$

$$Y = \alpha_1 e^{\beta_1 X_1} + \alpha_2 e^{\beta_2 X_2} + \varepsilon \quad (2.105)$$

which can not be transformed into linear equations and thus do not lend themselves to linear regression. The criterion used for determining the estimated values for coefficients is same as that used in linear regression, *i.e.* minimization of the sum of squared errors. Having N observations on X_1, X_2, \dots, X_k , the sum of squared errors is:

$$S = \sum_{i=1}^N \left[y_i - f(X_{1i}, \dots, X_{ki}, \beta_1, \dots, \beta_p) \right]^2 \quad (2.106)$$

The estimation techniques determine $\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p$ which are estimates of $\beta_1, \beta_2, \dots, \beta_p$ that minimize the sum-of-squared-errors S .

II.5.2.1 Computational Methods for Nonlinear Estimation

There are three general approaches to the solution of nonlinear estimation problem. Most numerical estimation methods involve one of the approaches or a combination of two of them [55].

The *Direct Search Method* evaluates the sum of squared errors for alternative sets of coefficient values. The values which result in a minimum are chosen as the estimates. Computational requirements prohibit the usage of this technique if number of parameters to be estimated is large.

The *Direct Optimization Method* estimates parameters by differentiating the sum of squared errors function with respect to each of its coefficients, setting these derivatives equal to zero, and solving the resulting set of nonlinear equations. Considering the error function of equation (2.106) the set of nonlinear equations to be solved are:

$$\sum_{i=1}^N 2 \left[y_i - f(X_{i1}, \dots, X_{ik}, \beta_1, \dots, \beta_p) \right] \frac{\partial f}{\partial \beta_i} = 0 \quad \text{for } i = 1, \dots, p \quad (2.107)$$

These nonlinear equations must be solved simultaneously. This approach is seldom applied directly due to computational difficulty.

The *Iterative Linearization Method* linearizes the nonlinear error function equation around some initial set of values. Then the ordinary least square technique is used to generate a new set of coefficient values. The nonlinear equation is relinearized using the new coefficient values and the ordinary least squares technique is again applied to generate new coefficient values. This iterative process is continued until convergence is attained.

This approach has certain advantages including computational efficiency. If the function to be fitted is reasonably approximated by a linear equation very few iterations may be necessary. A second advantage is that it provides clear guidelines for statistical tests that are usually applied only to linear regression. However, there is no guarantee that the method will converge in the first trial. It may be necessary to restart the method from several different initial points to reach the best solution.

In general tests available in linear regression such as t and F statistics are applicable to the final linearization performed in the iterative linearization method. However, the R^2 test can be applied in its conventional sense to the nonlinear regression:

$$R^2 = 1 - \frac{\sum_{i=1}^N \varepsilon_i^2}{\sum_{i=1}^N y_i^2} \quad (2.108)$$

where all the variables have been defined earlier.

II.6 Chapter Summary

This chapter reviewed econometric forecasting methods from the System Identification view point.

The Single-Step-Ahead (SSP) forecast and Multi-Step-Ahead (MSP) forecast were described and their mathematical formulation was reviewed.

The common linear ARX and ARMAX model structures in an input-output frame work were described. The system models of the most common nonlinear model structures, NARX and NARMAX were described. The predictor form of these models were described for performing single-step-ahead and multi-step-ahead forecasts. Neural

Networks as nonlinear model structures were introduced. Their single-step-ahead and multi-step-ahead predictor forms were described.

The most common estimation techniques were reviewed and the linear and nonlinear least-squares estimation procedures were described. Linear variants of linear least-squares estimation such as maximum likelihood estimation and instrument variables were also explained. Nonlinear least-squares estimation and backpropagation were described. Backpropagation is the most common training algorithm for NN model structures.

Finally, an overview of linear and nonlinear regression analysis was given. The Ordinary Least Squares (OLS) and estimation techniques in nonlinear regression were described.

CHAPTER III

LONG-TERM LOAD FORECASTING: A NONLINEAR METHOD

III.1 Introduction

Long-term load forecasting has been an area of interest in different scientific communities. Economists have applied econometric techniques in long-term load forecasting problems [74]. Engineers with experience in short-term load forecasting have applied Neural Networks (NN) with similar approaches to those used in forecasting the peak load demand in one-day-ahead forecasts [49]. Statisticians have emphasized the use of time series forecasting techniques [55]. An integrated long-term load forecasting system should be able to accommodate the modeling capabilities of all these methods. The forecasting model should be in an econometric framework to relate changes in energy consumption to changes in the underlying factors such as the economy, population and weather. The forecasting model should include some level of nonlinearity to capture the nonlinear components of the cause and effect relations in the system. Finally, the forecasting system should include a time-series component to capture the autoregressive effects between energy sales or peak load demand in consecutive years.

Usually in long-term load forecasting problems the historical data base includes annual data of different influential variables for a maximum of 25 years. This information generates approximately 25 distinct data points. However, any developed model should be tested against different model quality measures. Such measures include accurate model structure and parameter selection as well as forecasting capabilities [45]. As a result a good portion of the available historical data set can not be used directly in the model

development process. Moreover, data collected some 25 years ago can not be relied upon strongly. Also, some electric utilities have not followed consistent standards in collecting, recording and storing data. Thus further reducing the useful historical data set. Consequently, long-term load forecasting methods must cope with small historical data sets.

Traditionally, all the available historical observations have been included in the model development process. The model which reproduced the data with which it had been trained most accurately was chosen to forecast the future energy demand. These forecasts usually departed from the actual energy demand within a short period from their development. One of main reasons was the fact that future events were not simply a continuation of the past trends. The developed models had learned many historical events which had little or no chance to repeat in the future.

An alternative model building approach is to reserve a portion of the historical data set for model evaluation. The time period covered by this reserved portion of the historical data set is called the forecasting period, and no information contained in this period is used in the model development. The portion of the historical data set used in model development, called the estimation set, is further divided into a training set and a validation set. The validation set is also called the testing set. Information in the training set is used directly in estimation of model parameters, whereas information in the validation set is used indirectly in the model development. In particular, information in the testing set is used for out-of-sample testing, i.e. for cross validation, allowing prevention of over-estimation. Prevention of under-estimation and over-estimation is crucial in the development of a reliable forecasting technique and in particular in nonlinear estimation techniques. Figure 3.1 shows the different time periods selected in the developing a forecasting model.

Section III.2 proposes several objective functions to deal with unique characteristics of long-term load forecasting and discusses drawbacks and advantages of each one. Section III.3 introduces the Monte Carlo filtering process and defines its components in context of long-term load forecasting. Moreover, it presents the proposed

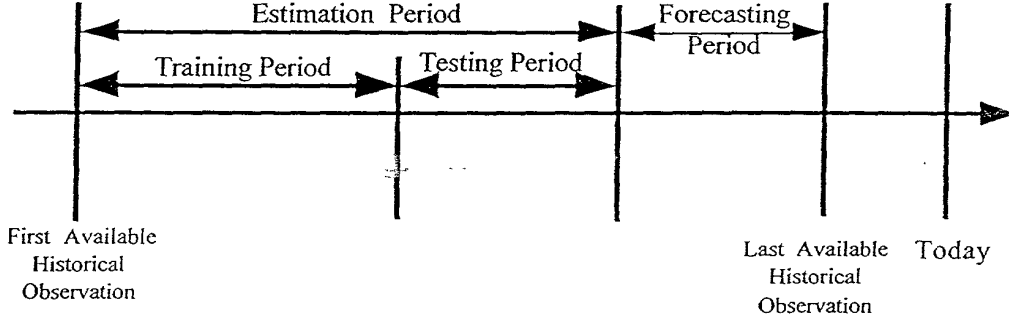


Figure 3.1: Different Time Horizons in Developing a Forecasting Model.

application of this method to optimize different modeling parameters in long-term load forecasting problem in detail. Section III.4 presents a flow chart of the proposed long-term load forecasting method. Section III.5 summarizes the chapter.

III.2 Objective Function Formulation

The historical data base of different relevant exogenous variables and the endogenous variable for a total period of N_t years can be written as:

$$\mathbf{Z}^{N_t} = [\mathbf{y}^{N_t}, \mathbf{u}^{N_t}] \quad (3.1)$$

$$\mathbf{y}^{N_t} = [y(1), y(2), \dots, y(N_t)]^T \quad (3.2)$$

$$\mathbf{u}^{N_t} = [\underline{u}(1), \underline{u}(2), \dots, \underline{u}(N_t)]^T \quad (3.3)$$

where \mathbf{u}^{N_t} is the set of the historical exogenous variables and \mathbf{y}^{N_t} is the set of historical annual energy sales for a particular class of customers or annual peak load demands.

Traditionally, all the available historical information has been directly used in developing the forecasting model. The model structure and parameter which reproduced the historical data which it had been trained with more accurately, has been chosen as the forecasting model. However, it is not possible to test the forecasting capability of the developed models using this approach.

We propose division of the historical information into two major subsets: the estimation set and the forecasting set. No information from the forecasting set is used in developing the forecasting model. Since the generalization capabilities of the developed models are crucial, the estimation set is further divided into the training set and the cross-validation set. Therefore:

$$\begin{aligned} \mathbf{Z}^N &= [\mathbf{y}^N, \mathbf{u}^N] \\ \mathbf{Z}^{E_N} &= [\mathbf{y}^{E_N}, \mathbf{u}^{E_N}] \\ \mathbf{Z}^{V_M} &= [\mathbf{y}^{V_M}, \mathbf{u}^{V_M}] \end{aligned} \quad (3.4)$$

where N , E_N , V_M are the lengths of estimation, training and validation sets respectively. Using the Neural Networks (NN) estimation methods discussed in the previous chapter a model is fitted to the training set:

$$\hat{y}(t|\theta) = g(\varphi(t), \theta) \quad (3.5)$$

where $\hat{y}(t|\theta)$ is the prediction of $y(t)$, $\varphi(t)$ denotes the data up to and including time t , where t is the last year in the training set and θ is the parameter vector. The procedure to obtain such a model considering different system models and estimation

techniques was explained in the previous chapter. Hence, we assume that a NN model is to be used as described in equations (2.25) to (2.39).

Now, assume that the observed data were generated from:

$$y(t|\theta_o) = g(\varphi(t), \theta_o) + e(t) \quad (3.6)$$

where $e(t)$ is a white noise error term with constant variance. Assuming a least squared error criterion for model parameter estimation, the objective function to be minimized is:

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^N \varepsilon^2(t, \theta) \quad (3.7)$$

where

$$\varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta) \quad (3.8)$$

Historically, such an objective function has been minimized over all the historical observations to obtain a forecasting model. This formulation is based on the assumption that future events are continuation of the historical trends, and modeling the historical events as good as possible guarantees accurate future forecasts.

Accepting the fact that future events are not continuation of historical events forces inclusion of a component to the objective function which accounts for the performance of the forecasting model on future unseen data. Therefore, we propose introducing some form of regularization to the objective function [66], [67]. The objective function will theoretically be modified to the following:

$$W_N^\delta(\theta, Z^N) = V_N(\theta, Z^N) + \delta \|\theta - \theta^\# \|^2 \quad (3.9)$$

The above equation denotes known parameters as $\theta^\#$. Lacking proper knowledge, $\theta^\#$ is replaced by a nominal guess.

Regularization can also be achieved using early stopping of the minimization search. This means that the nonmodified criterion (equation 3.7) is minimized by an iterative method, but the numerical search is terminated before the minimum is reached [67].

The key issue is when to terminate the minimization. This design question corresponds to the choice of δ in equation (3.9). In general, the number of iterations is chosen by reference to a cross validation set which is not used directly in the parameter estimation. This is natural since the minimization is the expected value of the objective function V_N . That is, the selected model should perform well on data which has not been used in the parameter estimation. If the objective function on the training set be and the validation set be expressed, respectively as:

$$V_N^E(\theta, Z_N^E) \quad (3.10)$$

$$V_M^V(\theta, Z_M^V) \quad (3.11)$$

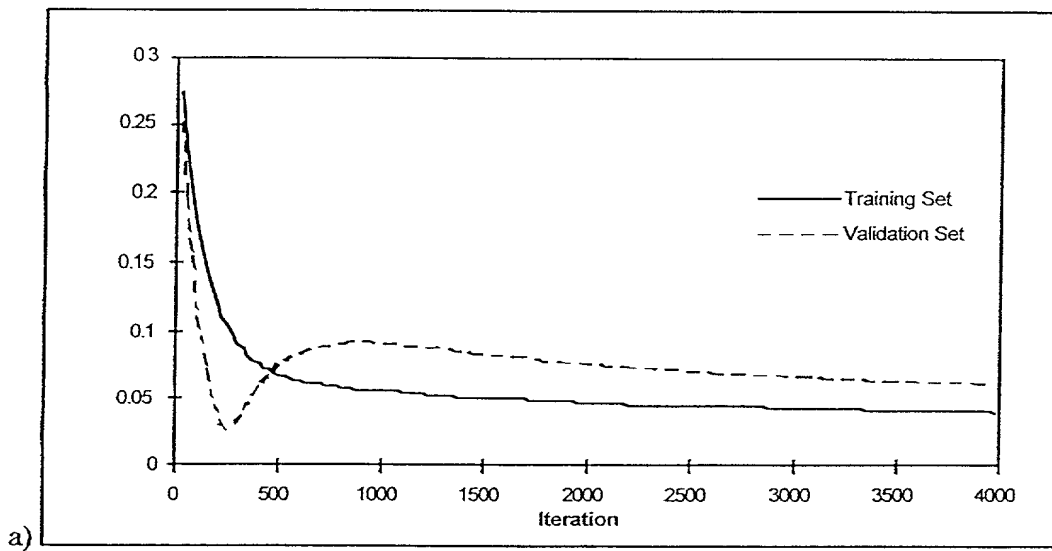
We suggest several stopping criteria for modeling long-term load forecasting problem, based on the relative value of these two functions, and discuss advantages and drawbacks of each stopping criterion. The first stopping criterion to be considered is:

$$V_M^\theta(\theta^{(i+1)}, Z_M^V) > V_M^\theta(\theta^{(i)}, Z_M^V) \quad (3.12)$$

The training process continues as long as the performance on the validation set improves and stops as soon as further training deteriorates the model performance on the validation set. In this formulation the training stops based on the validation set, that is, when the validation set error reaches a minimum.

Figure 3.2 shows two examples of error profiles in the above formulation where the data set corresponds to the historical data set found in long-term forecasting

problems. It is common to encounter situations where the validation set error drops to a minimum while the training set error is still large. This situation is commonly called underfitting in the NN community. It is also common to encounter situations where the initial parameter values perform better on the validation set than on the training set. Therefore, the training set error remains larger than the testing set error for a large number of iterations. Therefore, the use of the stopping criterion of equation (3.12) may result in selection of undesirable network models, particularly for limited data sets such as those found in long-term load forecasting. The advantage of such stopping criteria is



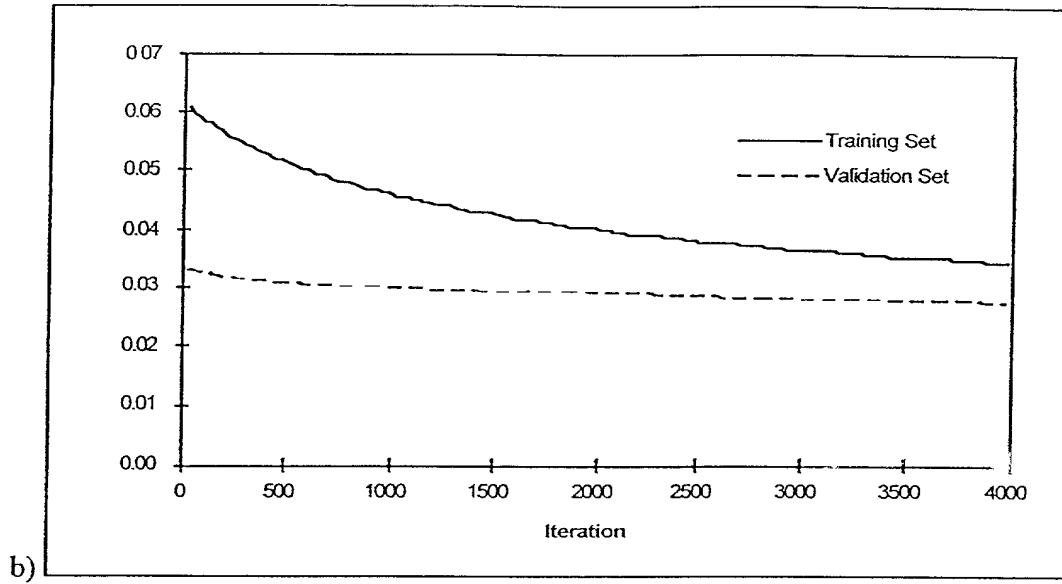


Figure 3.2: Behavior of the Training and the Validation Set Error Profile When the Stopping Criterion Is As in Equation (3.12): a) Underfitting b) Not Crossing.

that the estimation process stops without the interference of the forecaster in setting any tolerance level on the number of iterations.

A second stopping criterion can be considered as:

$$\begin{aligned} V_N^E(\theta, Z_N^E) &= V_o \\ V_M^V(\theta, Z_M^I) &= Low \end{aligned} \quad (3.13)$$

Using this stopping criterion the forecaster would select the value V_o at which to stop the training process. Deciding on the proper error level of the training set requires some experimentation with different error levels. Stopping at large training error levels will result in large errors in the training set. In the NN community this is referred to as

underfitting. Terminating the training at small training errors may also result in large validation error. In the NN community this is referred to as overfitting. Figure 3.3 shows examples of overfitting and underfitting. At a certain iteration in the training process the error in the validation set drops to a minimum. However, we have no information about the required error on the training set to achieve low validation set error. Setting the training set error to a fixed value does not guarantee that all the networks with similar performance on the training set will have similar performance on the validation set. Finding an adequate number of networks with low error levels on the validation set usually requires several hundred weight and network initializations.

A third stopping criterion can be proposed as:

$$V_N^E(\theta^{(i)}, Z_M^V) = V_M^V(\theta^{(i)}, Z_M^V) \quad (3.14)$$

The estimation process stops at a point at which the training set and the testing set errors equal. In another words, the developed models learn the historical information to the same level at which they perform in future forecasts. Thus, at the stopping point, the developed models forecast future energy sales with the same accuracy as they

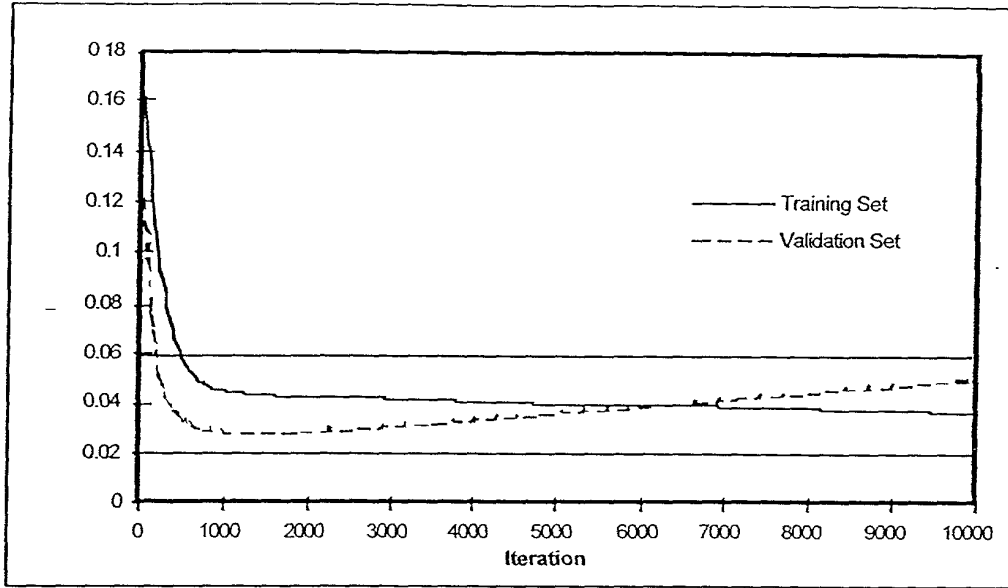


Figure 3.3: Behavior of Training and Validation Set Error Profile When the Stopping Criterion is as in Equation (3.13).

reproduce the historical energy sales.

We used an iterative learning algorithm to implement this proposed stopping criterion. At each iteration the error on the training set and the validation set is computed. Models which result in training set and the validation set error difference below a small tolerance are selected.

Obviously, the advantage of this stopping criterion is its independence from forecaster actions in terminating the estimation process. However, several observations can be made, regarding this stopping criteria (3.14):

1. In many occasions the training set and the testing set error profiles will remain apart and do not cross at any point during the estimation process.

2. In networks where the training set and the testing set error profiles cross, there can be instances of multiple crossings. In these cases the lower crossing error levels should be selected.
3. Random initialization of network parameters (weights) may result in crossing points at different error levels, even if the number of network parameters is fixed.
4. Networks with different numbers of parameters (weights) may have the crossing point at different error levels.

Clearly, the forecaster and model designer must be aware of all the above conditions in the model building process. Figures 3.4 and 3.5 illustrate several of above observations.

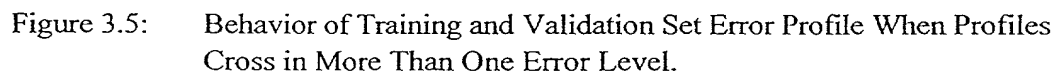
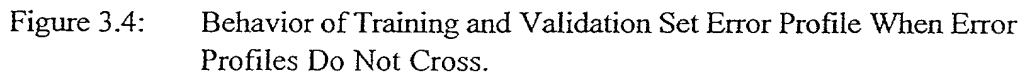
After discussing the selection of the stopping criterion, we discuss other factors in designing a nonlinear long-term load forecasting system.

In linear estimation, correlation tests reveal the autocorrelation and cross-correlation among annual energy sales of consecutive years, as well as between annual energy sales and each of exogenous variables. However, no similar tests exist in nonlinear estimation. NN is a nonlinear modeling technique. The most reliable method in identifying the level of the autocorrelation and cross-correlation is through simulation. The performance of models developed using different combination of autocorrelation and cross-correlation terms can be used to select the most influential historical exogenous and endogenous variables.

The number of parameters in any model is closely related to the number of data points and the dynamics of the problem. As the number of data points increases, repeating the estimation process will result in consistent estimates. The ratio of number

741

of data points to model parameters is an indication of consistency in obtained estimates. Large ratios guarantees a consistent estimate. However, as the ratio decreases to close to



unity, repetitive trials generates inconsistent model parameters. In general, as the number of points and dynamics of the problem increase, more parameters are required to generate consistent estimates.

The consistency of estimates is a crucial factor in long-term load forecasting due to the scarcity of the available data points. At maximum, training set includes 15 points of annual data. Data points include stochastic components, since they are different geographic, social and economic indicators. Small number of data points limits the size of feasible network structures to include only few parameters. Small NN models can be built by considering only networks with single hidden layer and only few number of hidden nodes.

Generally, models with small parameters may perform equally good on the training set. However, there is no guarantee about their similar performance on validation set or forecasting set.

Above observations reveal that to find a number of forecasting models with good performance on historical as well as future data set, different model structures as well as model parameters should be searched. The search must account for all above considerations.

We propose the application of Monte Carlo filtering process in the selection of a number of candidate forecasting models. Monte Carlo filtering method has previously been used in identification of influential indicators in environmental studies [34], [37], [60]. However, no application to long-term load forecasting has previously been published.

III.3 Monte Carlo Filtering

Historically, data bases spanning several decades are not always available for mathematical. In many cases, the requisite data are extremely difficult, expensive or even impossible to obtain. In such cases traditional, statistically based methods for estimating model parameters cannot be applied. In the environmental community, models for this class of processes have been categorized as *speculative simulation methods* since the available data, or at least portions of the data, are so sparse that rigorous validation of the model is not possible [37].

Monte Carlo filtering [34], [60], the process of rejecting sets of model simulations that fail to meet prespecified criteria of model performance, is a useful procedure for objectively establishing parameter values and improving confidence in model predictions. This method involves: 1) the use of available information or data to establish the set of acceptable behaviors, 2) application of Monte Carlo methods to vary parameters over a range of values and generate corresponding sets of model performance indices, and 3) classification of each simulation as acceptable or unacceptable according to the prespecified behavior definition. The subset of models and their parameters that generate acceptable model simulations can be regarded as equally satisfactory models. The set of acceptable models can then be used to make predictions about future values of the variable of interest.

Monte Carlo filtering can be performed in a deterministic or stochastic framework. In the deterministic framework variables are selected from a prespecified set of acceptable values. In the stochastic form, variables are selected randomly from prespecified acceptable distributions. Each set of parameter realization generates a single performance index. The realization is classified as acceptable if the performance index meets acceptable behaviors conditions. Performing Monte Carlo simulations for a large

number of times results in collection of an ensemble of realizations which satisfy acceptable performance conditions. The sets of acceptable parameter values are also useful in identifying the most important parameters within the available set of parameters in the simulation.

Given historical observations of annual energy sales and possible influencing variables such as annual number of customers, price of electricity, CDDs and HDDs, the result of proposed nonlinear long-term load forecasting method using Monte Carlo filtering is identification of the most influential autocorrelation and cross-correlation terms, number of hidden nodes, and model parameters (weights). The selected realizations are then used to forecast annual energy sales, or annual peak load demand.

The search for the best combination of autocorrelation terms, cross-correlation terms and hidden nodes is a deterministic search. This is performed by searching the acceptable range of these variables. As in every deterministic simulation, these sets of parameters are changed in steps to cover all different possible combinations. The weight initialization is a stochastic search. Weights (or model parameters) are initialized from a uniform distribution. In general, limits of this range are +1 and -1. The number of initializations depends on many factors among them the type of data which the forecaster is trying to model. The above procedure describes a mixed deterministic and stochastic Monte Carlo filtering technique.

Extremely short historical time series forces the search among different autocorrelation terms, cross-correlation terms and hidden nodes to be confined to few numbers. At each combination of these variables, several hundred weight initializations is performed. Training of each model is done using a NN learning algorithm. At each iteration, training error, validation error and their difference is computed. The training continues for a large number of iterations. The network parameters corresponding to the

iteration with the smallest error difference is stored. If this error difference is within a small tolerance level, the obtained network parameters satisfy the condition of acceptable behaviors. Network with acceptable error level will be selected and ranked after several hundred networks have been trained. A predetermined number of these networks will be selected at the end of each deterministic search iteration, and their average error and variance will be computed. These forecasting models have the same autocorrelation terms, cross-correlation terms and hidden nodes.

Above procedure is performed for every combination of autocorrelation terms, cross-correlation terms and hidden nodes specified in the deterministic search range, and the corresponding average error and variance of selected networks in each model structure is computed. These model structures are ranked based on the lowest average error and variance of their selected networks.

A predetermined number of top performing model structures are selected to forecast the annual energy sales or annual peak load demand in the forecasting period. The final forecast will be an average of these forecasts. The selection of best model structure and model parameters is based on the historical information and information in the forecasting period is only utilized in final forecast.

III.4 Flow Chart of Monte Carlo Filtering Process

Monte Carlo filtering process includes training thousands of networks within the feasible region of parameters. Networks and structures which satisfy several statistical measures based on the training and the validation set will be selected. The selected networks will be trained, and used in the final forecast.

The computational time required to develop a single forecasting model using this approach requires about an hour of computational time on the currently available workstations. This time is mainly required for initial network search in the stochastic search.

Figures 3.6 to 3.10 show the flow chart of the proposed nonlinear forecasting system using Monte Carlo filtering process. An explanation of important variables and modules of the program are given as:

flag1	1	Trains a NN and stores the weights.
	2	Trains a NN with the validation set, and stores the weights based on the minimum training and the testing error difference.
	3	Passes a set of inputs through the network and computes the output.
flag2	1	Uses the testing set sequentially.
	2	Uses the testing set, randomly.
flag3	1	Uses the present information to predict the output.
	2	Uses the past information to predict the output.
flag4	1	Uses the <i>tanh</i> as the discrimination function.
flag5	1	flag6 = 1 Standard Backpropagation (batch update) algorithm is used.
	2	flag6 = 1 Adaptive Backpropagation algorithm.
flag6	1	Batch Update of the weights.
	2	Individual Update of the weights.
flag8	1	Uses linear sum as the output discrimination function.
	2	Uses <i>tanh</i> as the output discrimination function.

flag9	0	No weight decay technique is used.
	1	Weight decay technique is used.
flag11	1	Random initialization of weights is performed.
	2	Weights are read from the data file.
flag12	1	Error is computed as RMS error.
	2	Error is computed as variance normalized MSE.
	3	Error is computed as normalized RMS.
nip		Number of past inputs in the architect search.
nip_init		Initial range in the past number of inputs.
nip_fin		Final range in the past number of inputs.
nop		Number of past outputs in the architect search.
nop_init		Initial range in the past number of outputs.
nop_fin		Final range in the past number of outputs.
no_gf		Number of global feedbacks in the architect search.
gf_init		Initial range in number of global feedbacks.
gf_fin		Final range in number of global feedbacks.
no_er		Number of past error terms in architect search.
er_init		Initial range in the past error terms.
er_fin		Final range in the past error terms.
hidnode		Number of hidden nodes in architect search.
hidnode_init		Initial range in number of hidden nodes.
hidnode_fin		Final range in number of hidden nodes.
out_set		Number of architectures used in the final forecast.

749

max_run

Number of weight initializations in one architecture.

```
pri_out
```

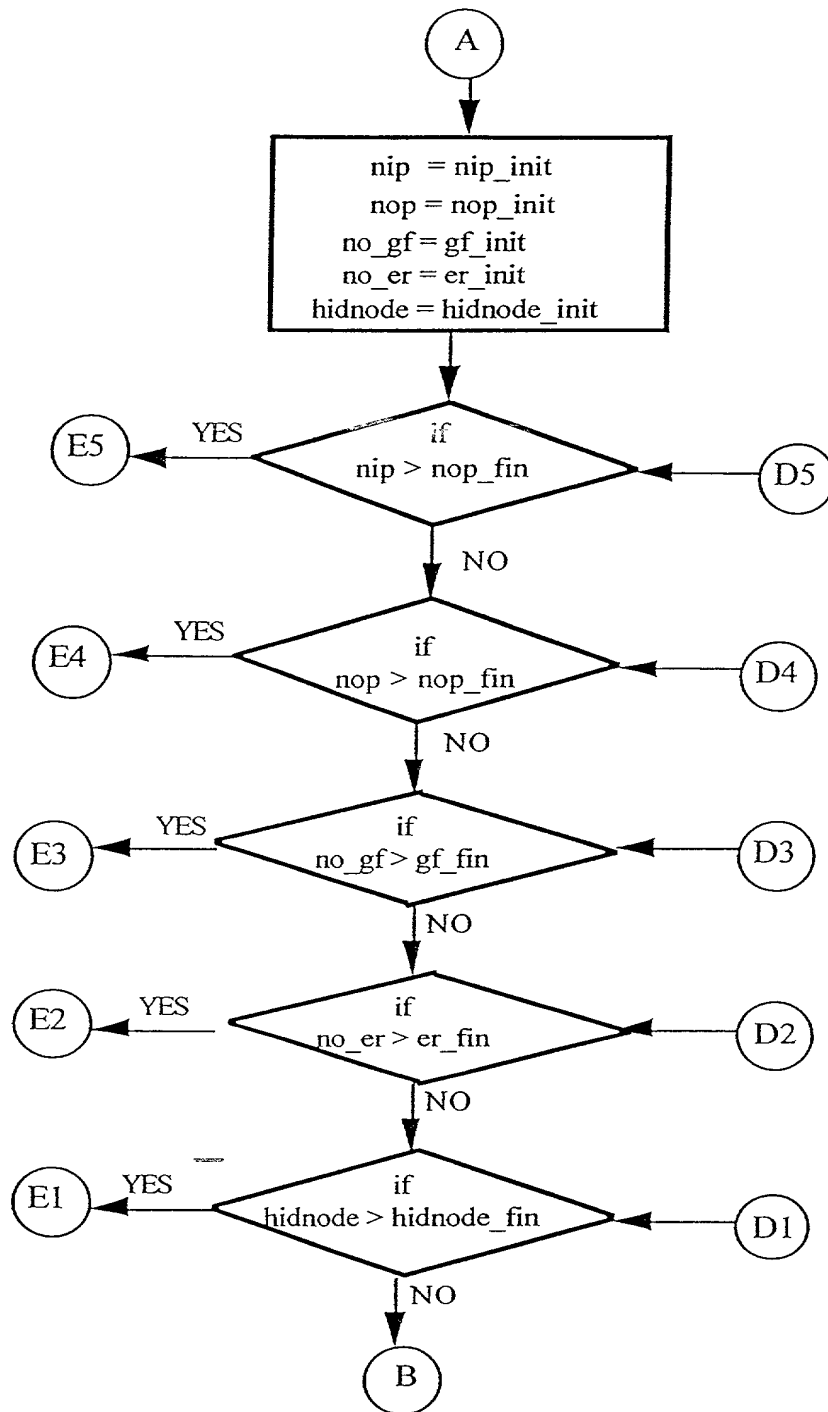
Number of networks in each architecture used in forecast.

final

Figure 3.6: Flow Chart of Long-Term Load Forecasting System.

Figure 3.6: Flow Chart of Long-Term Load Forecasting System.

752



66540 665660

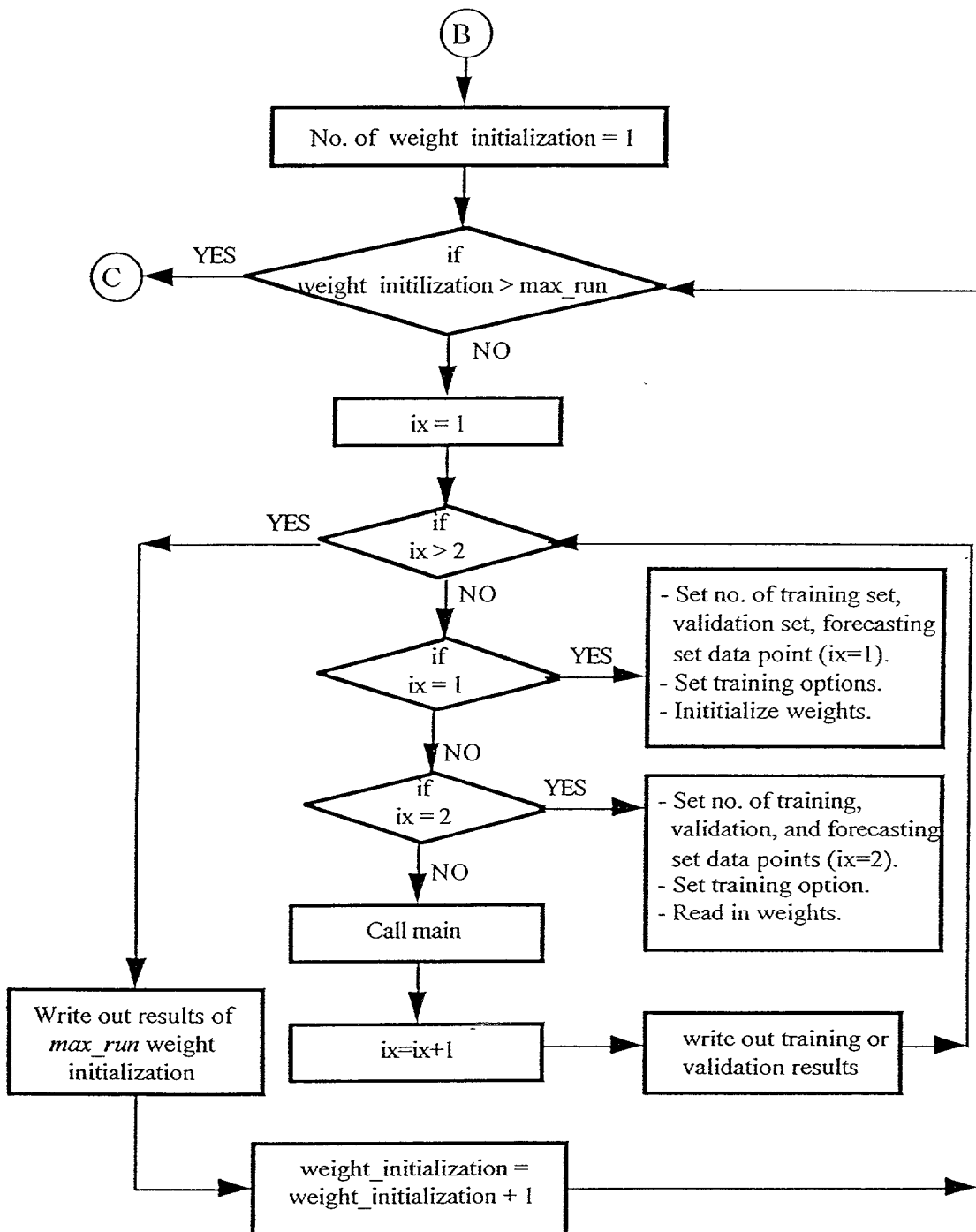


Figure 3.6: Continued.

754

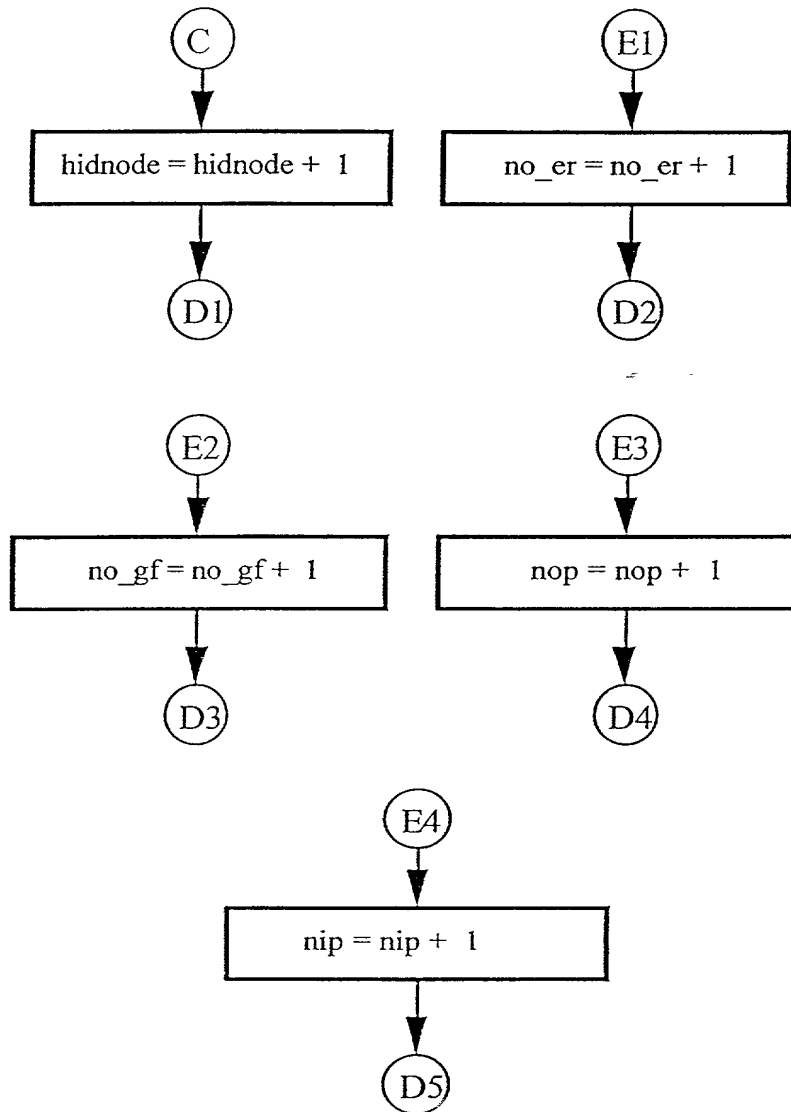


Figure 3.6: Continued.

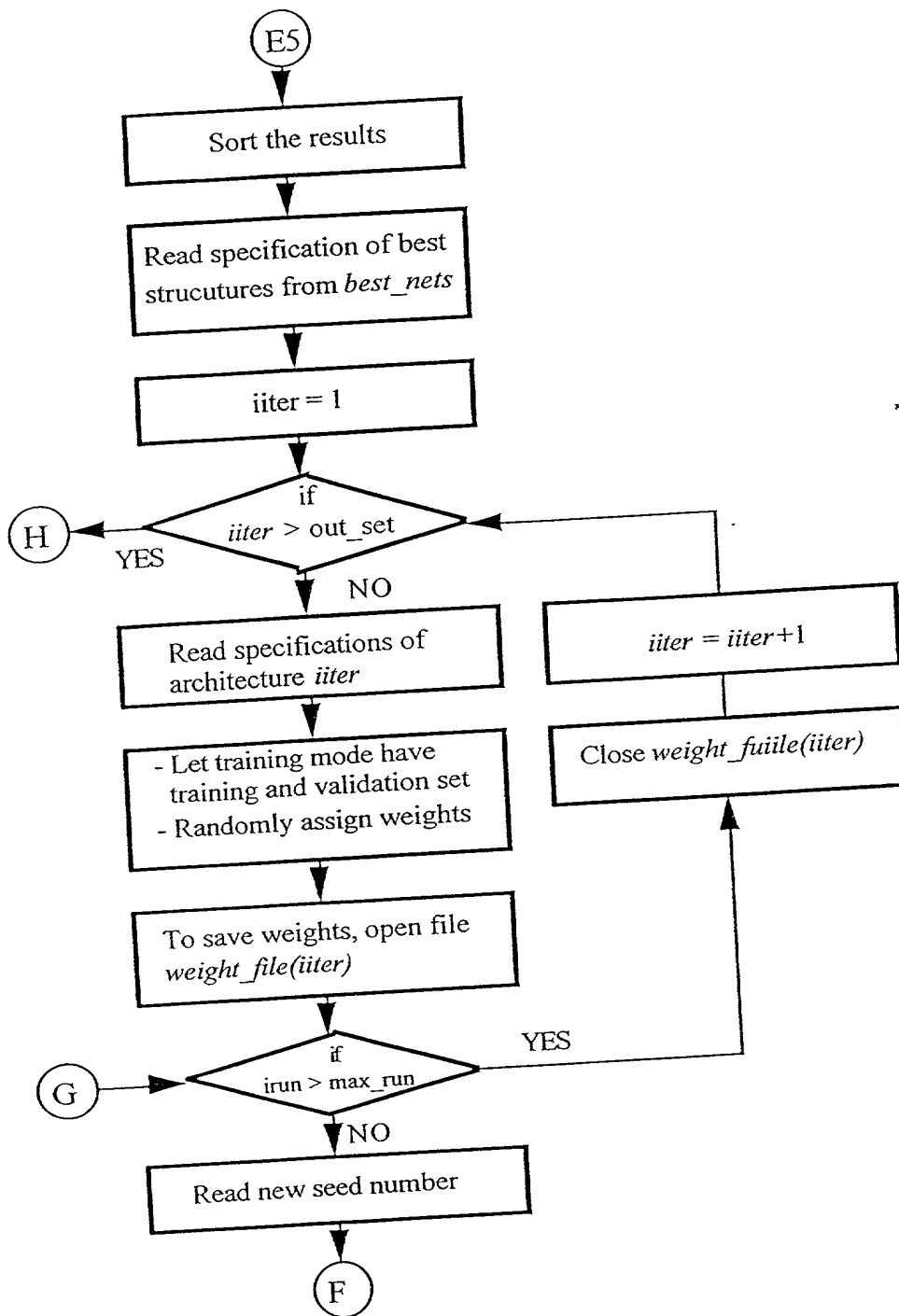


Figure 3.6: Continued.

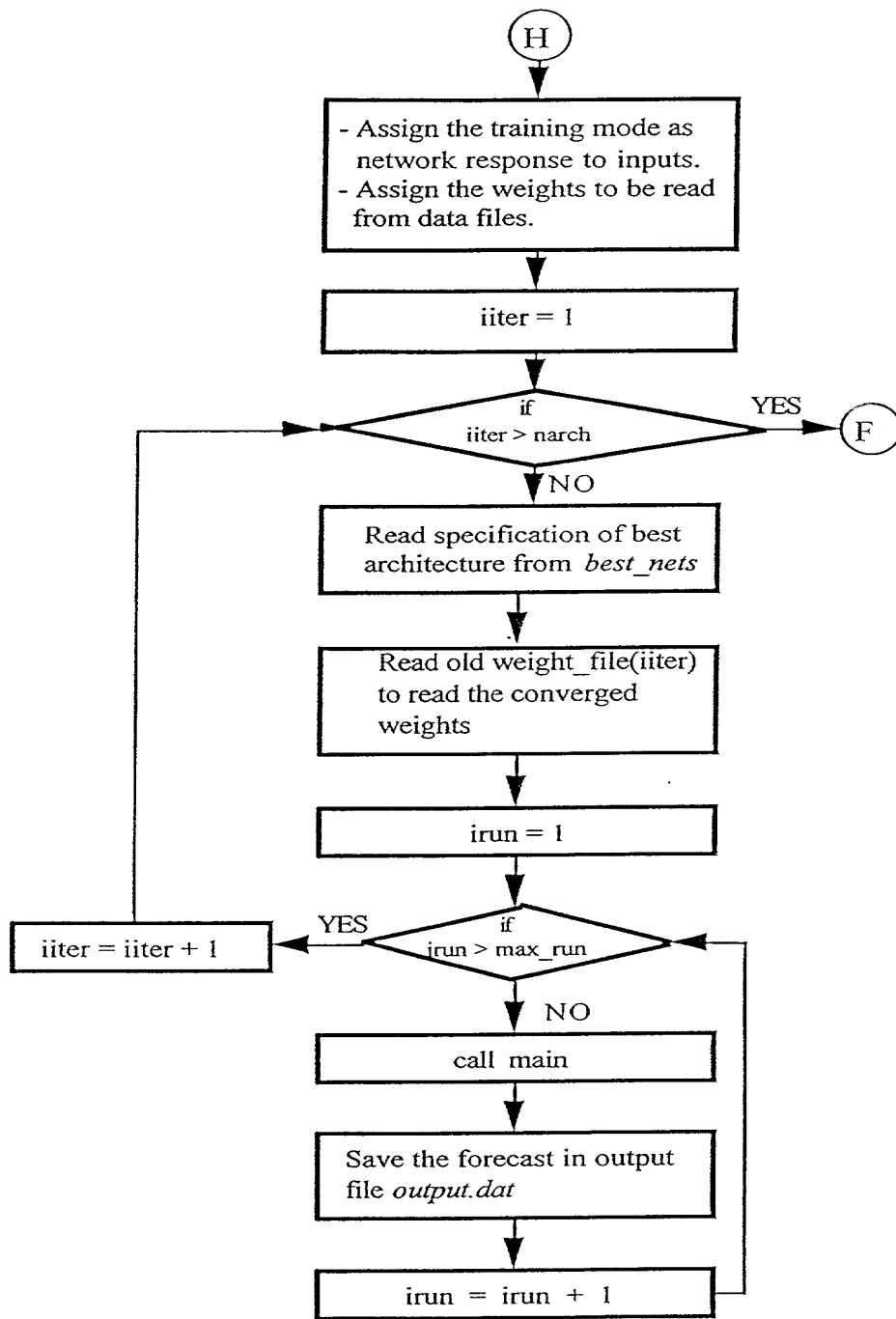


Figure 3.6: Continued.

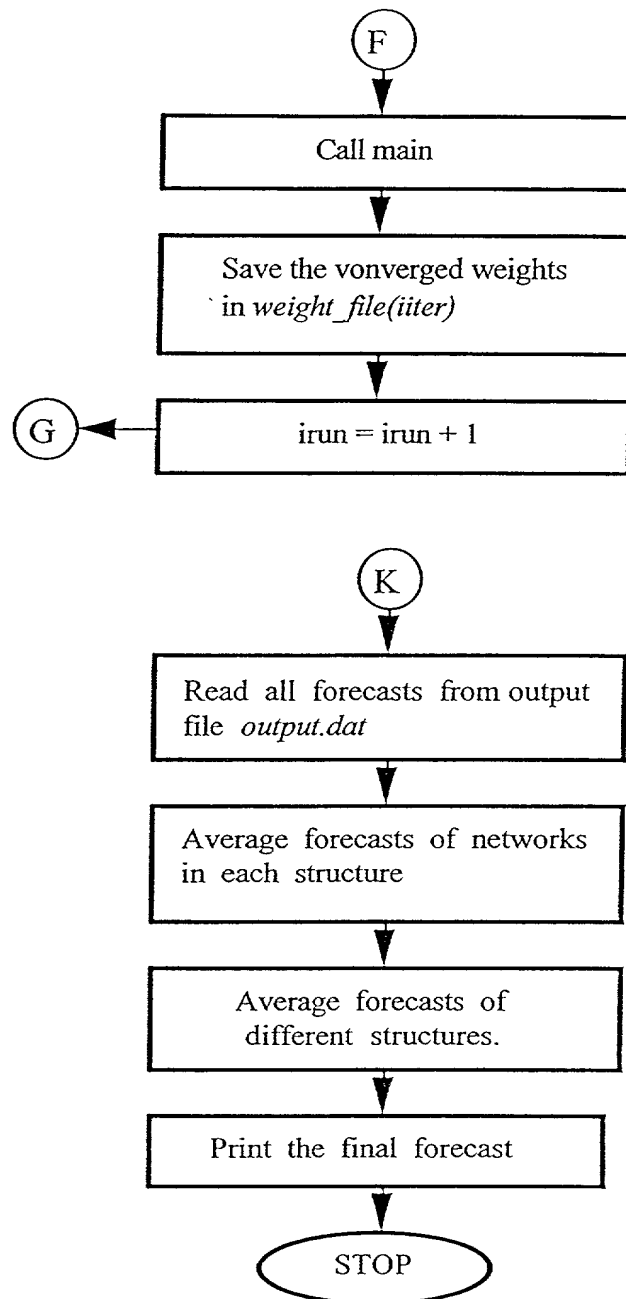


Figure 3.6: Continued.

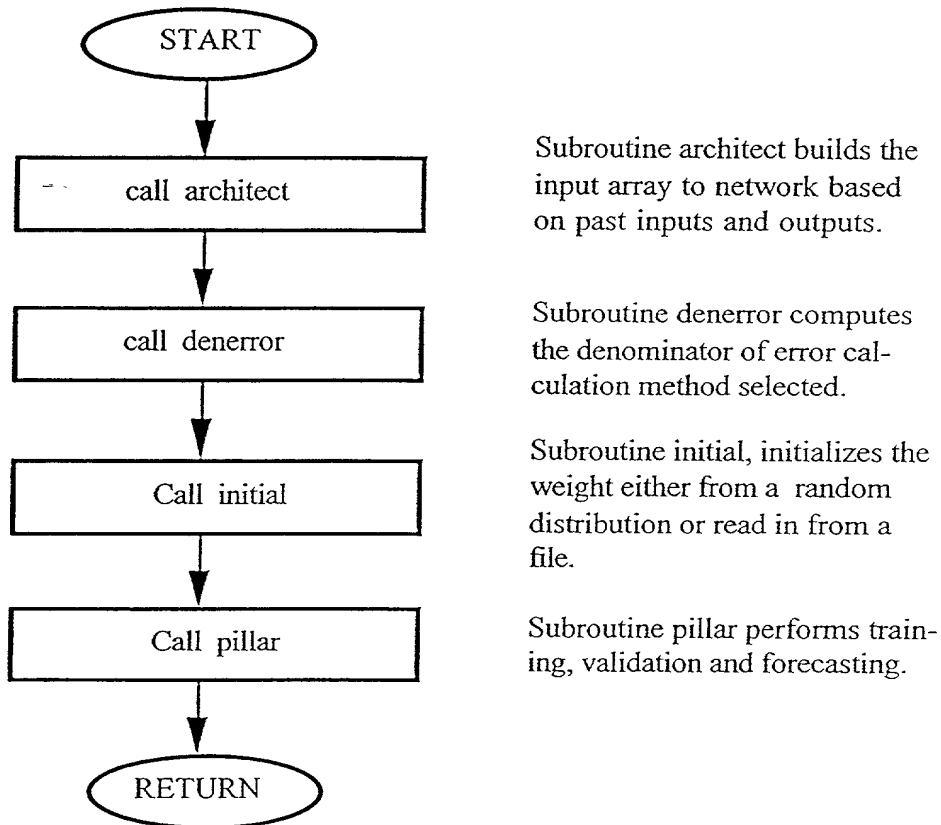


Figure 3.7: Flow Chart of Module *Main*.

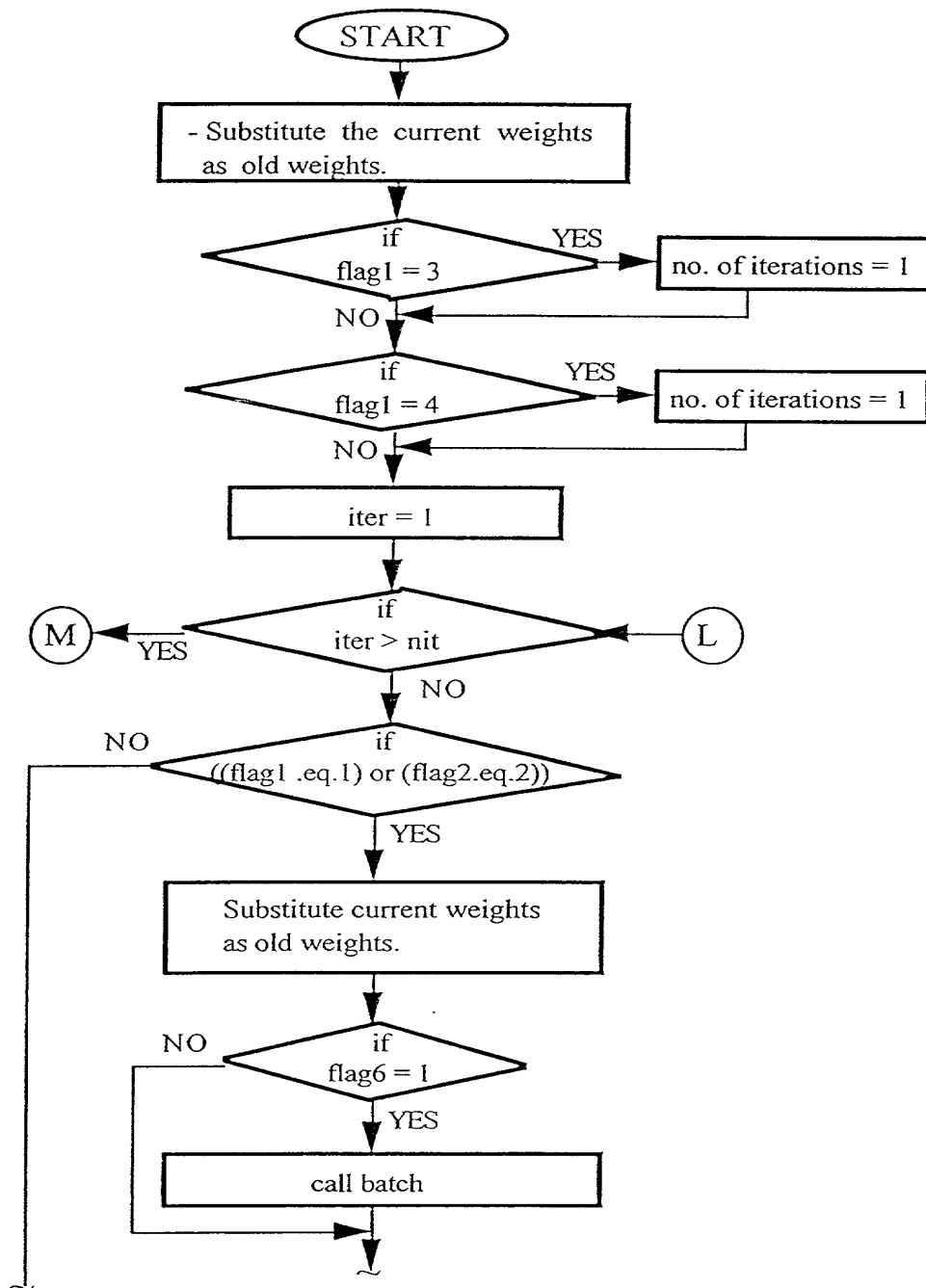


Figure 3.8: Flow Chart of Module *Pillar*.

760

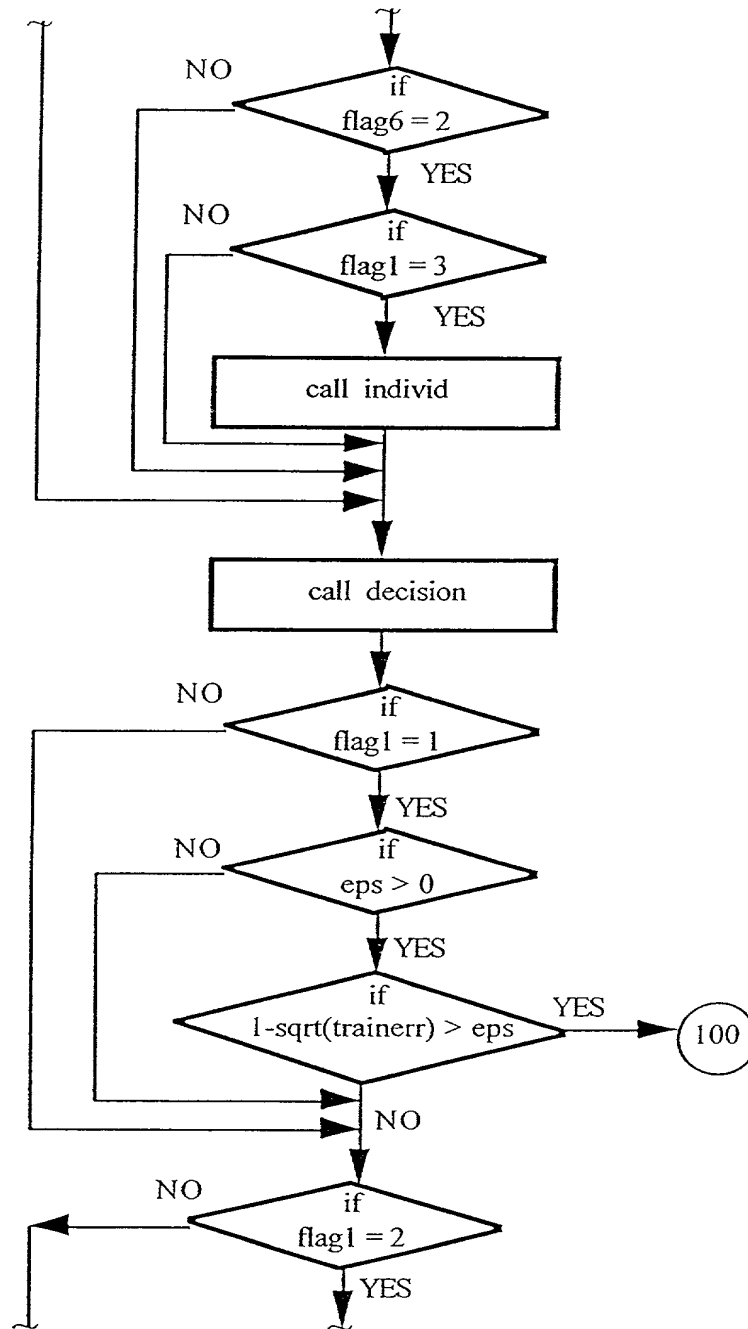


Figure 3.8: Continued.

761

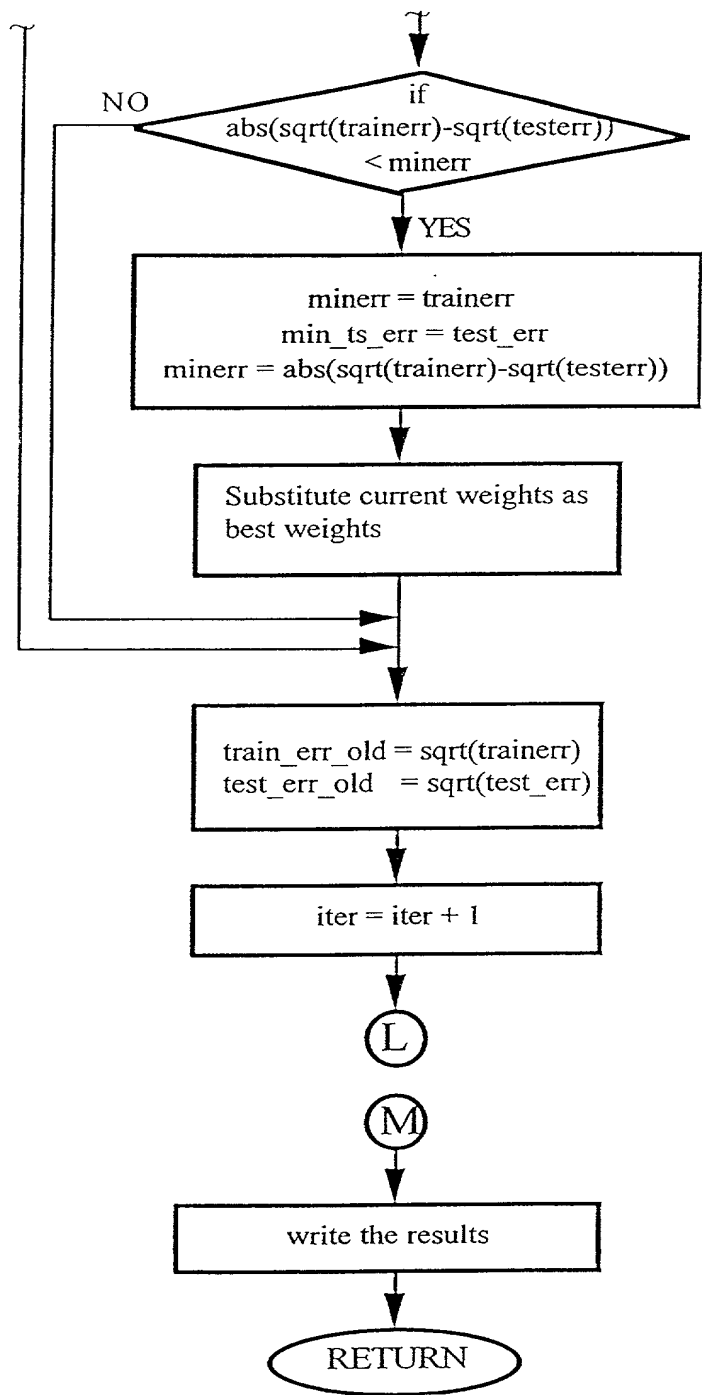


Figure 3.8: Continued.

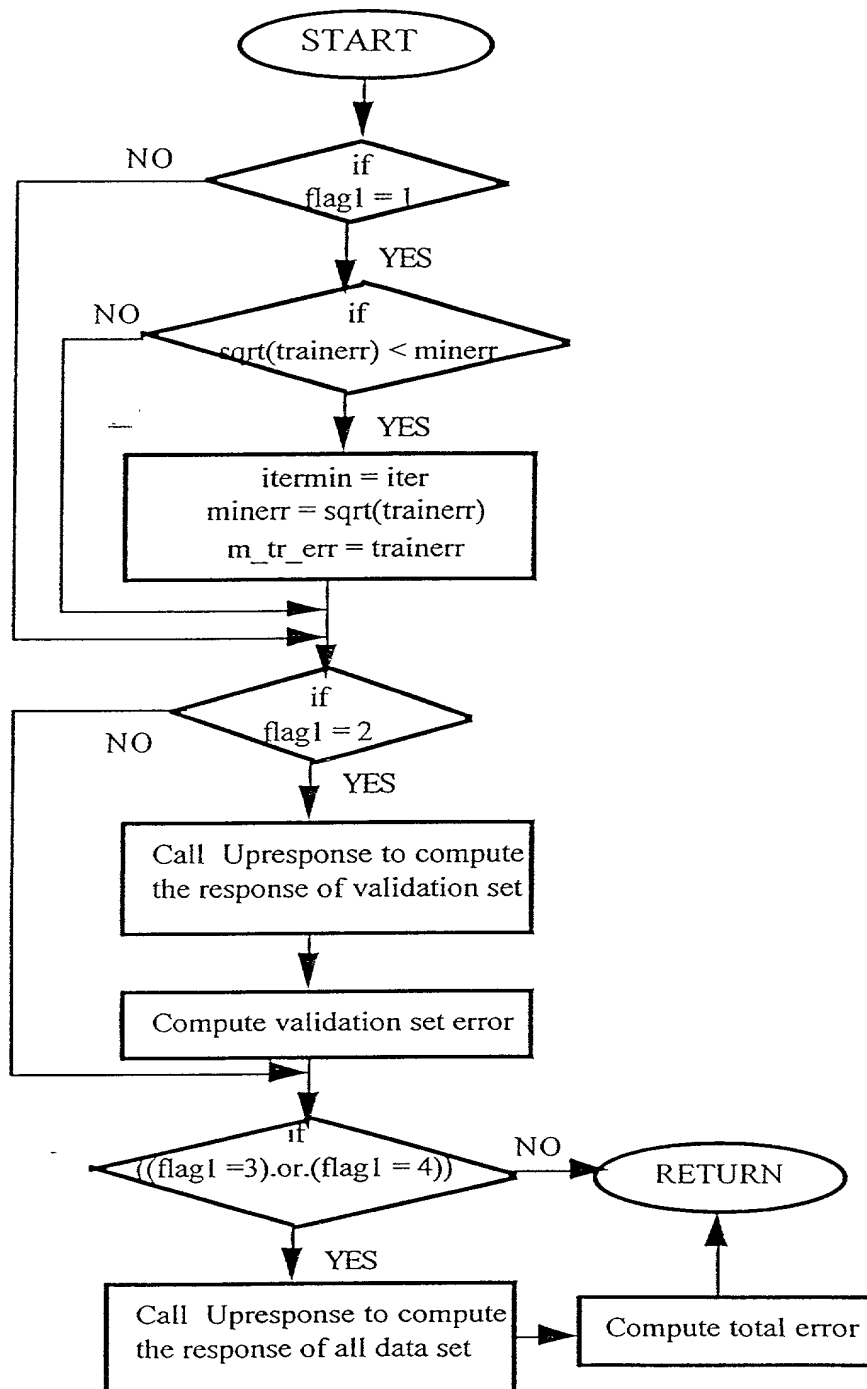


Figure 3.9: Flow Chart of Module *Decision* to Process the Validation Set.

763

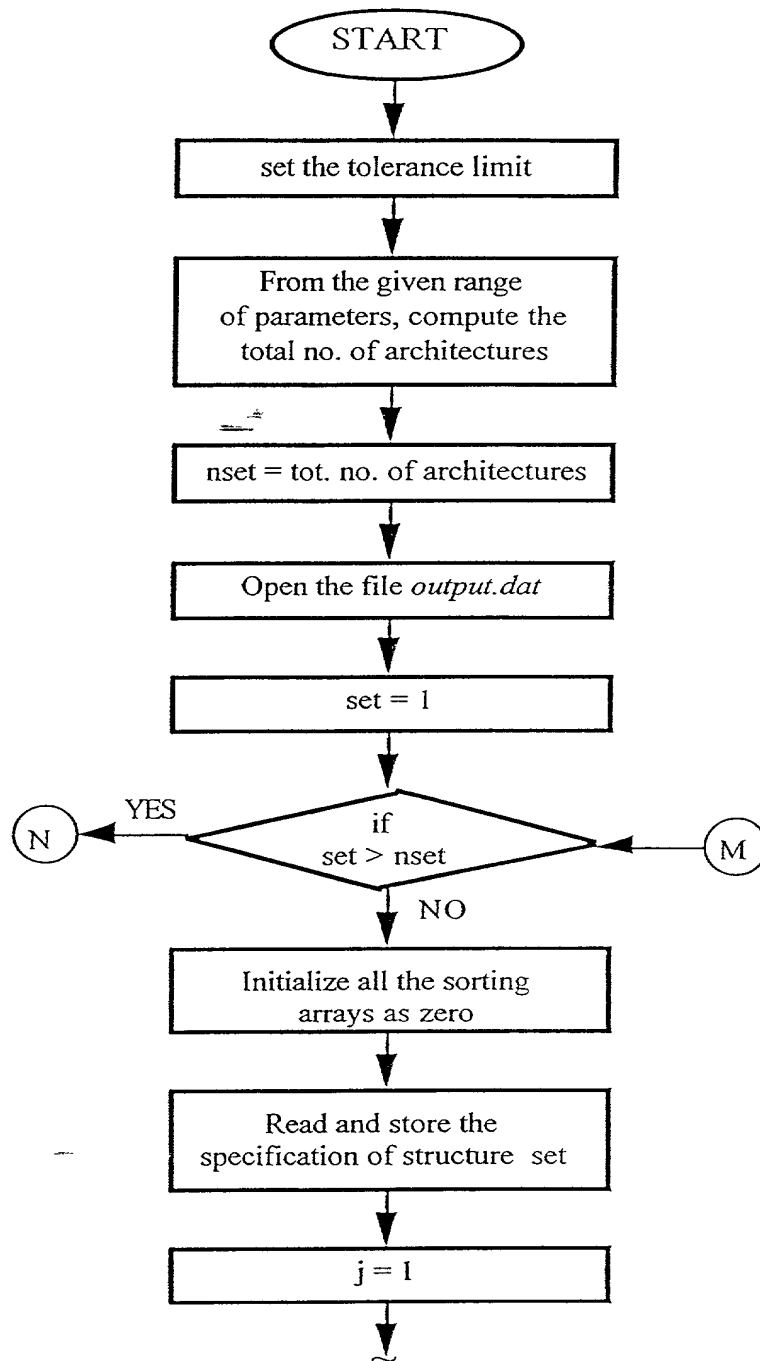


Figure 3.10: Flow Chart of Module *Sorting* to Sort the Results.

764

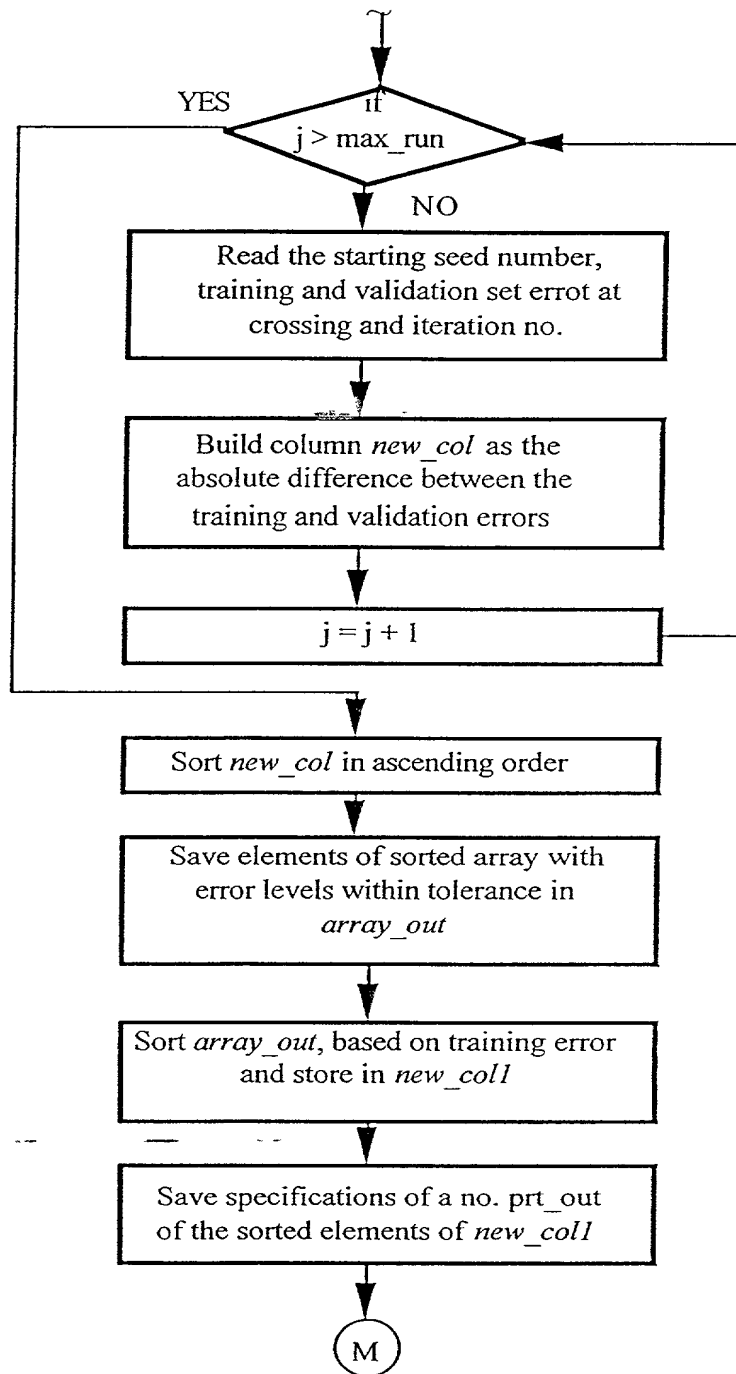


Figure 3.10: Continued.

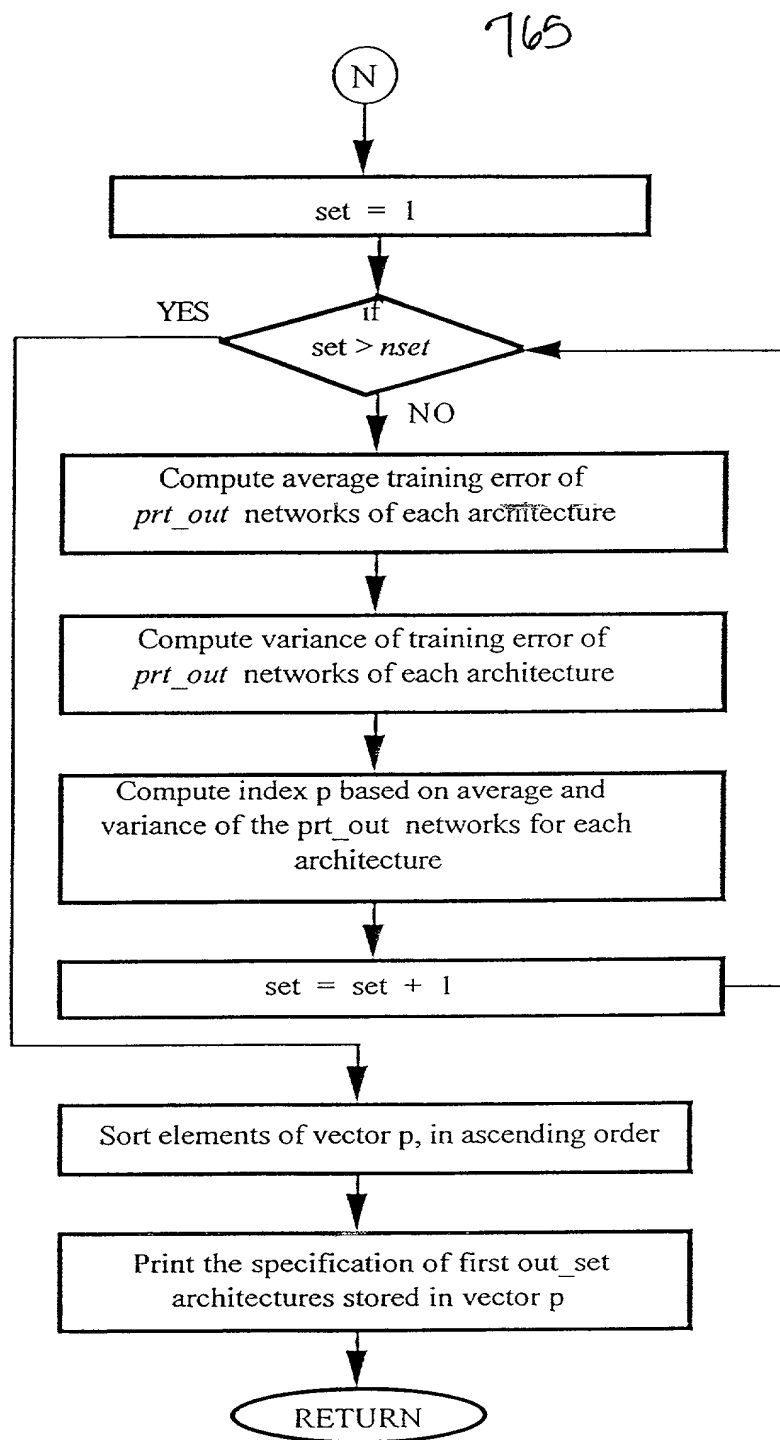


Figure 3.10: Continued.

III.5 Chapter Summary

In this chapter, several new concepts in long-term load forecasting were introduced. The available historical information was divided into two sets, instead of a single data set. The available historical information was divided into estimation set and forecasting set. The forecasting set was reserved to test the quality of the developed long term load forecast methods. The information in the forecasting set was not used in the model development process. The estimation set was further divided into a training set and a validation set. While, both sets were used in the model development process, only the information in the training set was directly used in parameter estimation.

A nonlinear forecasting technique was introduced. Several proposed parameter estimation criteria were reviewed and their advantages and drawbacks in the context of long-term load forecasting were discussed. It was shown that due to the scarcity of data in long-term load forecasting, more than one model can perform equally good on the training set. However, their forecasting performance may be quite different. A stopping criterion, based on the equal error levels on the training and the validation set, was proposed for modeling the long-term load forecasting problem. Several observations in using this stopping criterion were made such as multiple crossing points, and non-crossing of the training and the validation set.

Different model structures reveal different forecasting performance. To select a number of candidate model structures, the feasible space of model parameters and model structures must be searched. A Monte Carlo filtering process was proposed to select the candidate model structures and model parameters from the feasible region. This formulation included both a deterministic and stochastic search. Models satisfying predefined acceptable behavior are selected and used to forecast energy sales or peak load demand.

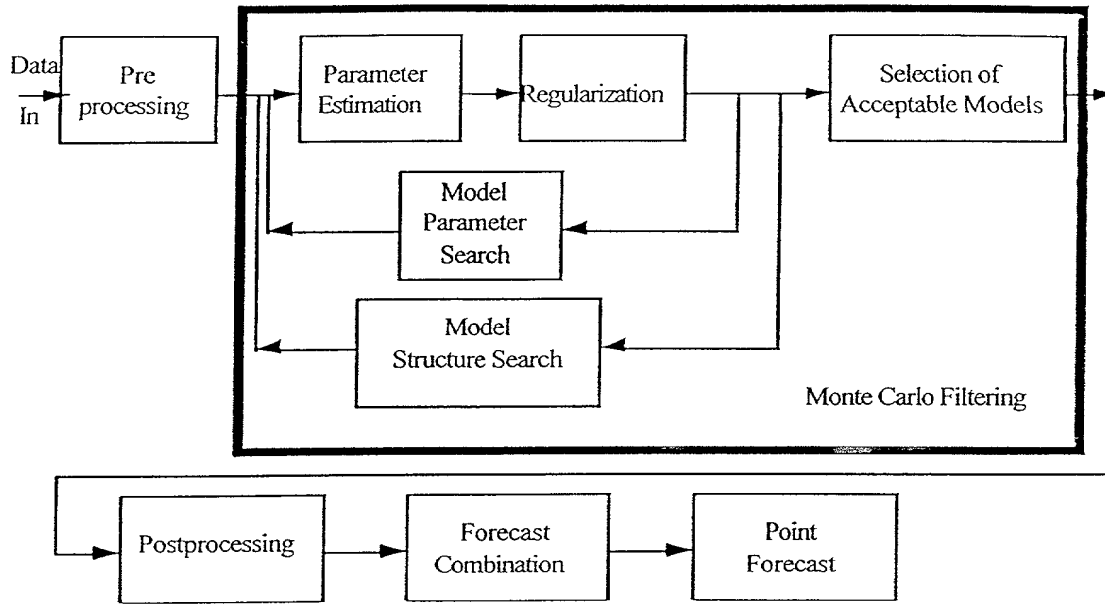


Figure 3.11: Block Diagram of the Long-Term Load Forecasting System.

A flow chart of Monte Carlo filtering process for nonlinear econometric long-term load forecasting was presented.

Figure 3.11 shows the block diagram of the overall long-term load forecasting system. Different relevant exogenous variables and energy sales or peak load demand are preprocessed using appropriate scaling factors. Monte Carlo filtering process generates an ensemble of model simulations. Each model simulation generates a performance index. The acceptable forecasting models are selected based on classification of the performance index. The forecasts from this submodel are combined to generate the point forecast of energy sales of peak load demand.

CHAPTER IV

FORECASTING MODEL DEVELOPMENT

IV.1 Introduction

This chapter presents the application of the nonlinear forecasting techniques developed in chapter III to forecast the energy sales and peak load demand in four Texas utilities which are members of Texas Municipal Power Agency (TMPA). The four utilities serve City of Bryan, City of Denton, City of Garland and City of Greenville.

This chapter is organized as follows: Section IV.2 presents the forecasting objectives and briefly explains the different exogenous variables available for forecasting the energy sales and peak load demand of the four utilities. Section IV.3 presents some general information on the total energy demand in TMPA and the share of the total demand for each of the member utilities. Section IV.4 presents a detailed analysis of the developed forecasting models for energy sales and peak demand of different customer classes as well as system peak demand for the City of Bryan. Section IV.5 presents a summary and comparison of the models developed for the other utilities studied. Section IV.6 summarizes the chapter.

Portions of this chapter have been published in an earlier report to the American Public Power Association (APPA) [51].

IV.2 Modeling Objectives

The nonlinear neural network based load forecasting methods developed in Chapter III have been used to forecast energy sales and peak load demand in the member utilities of TMPA. The forecasts using these new modeling technology are compared with forecasts made using other methodologies by the member utilities. Assessment of accuracy of forecasts have been made by backcasting using the historical data series provided by the utilities. In backcasting, that portion of the historical data series used for forecasting assessment is not used in any way in the model development process. The historical data series provided by the utilities consisted not only of actually observed loads and exogenous variables, but also forecasts of the exogenous variables at certain points in time as used in previous load forecasts by the utilities. These data permit both conditional and unconditional backcasting assessment of model accuracy and comparison of accuracies between the new methods developed in the current study and the methods used in the past by utilities.

In this chapter, the terms “backcasting” and “forecasting” are used interchangeably and refer to forecasting energy sales and peak load demand within a horizon of the historical period.

Exogenous variables in econometric long-term load forecasting may include the number of customers in the service area, total population, number of households, Cooling Degree Days (CDD), Heating Degree Days (HDD), real or adjusted price of electricity, Per Capita Income (PCI), and Gross National Product (GNP). The CDD is a temperature variable and is proportional to air conditioning load of the service area. It is proportional to the number of days per year on which the average daily temperature is above 65°F. Similarly, HDD is proportional to the heating load and is proportional to number of days on which the average temperature is below 65°F. In the early 1970's and

before, the difference between the monthly average temperature and 65°F was assumed to exist in all the days in the month. Therefore, each month could have either CDDs or HDDs. However, as more computing facilities became available, the CDD of each day was computed based on the average daily temperature. Using this formula, it is possible to have a number of CDDs and HDDs within a single month. The adjusted price of electricity is the actual price of electricity normalized by the Consumer Price Index (CPI) where the Consumer Price Index can be on a state or national level. Per Capita Income (PCI) is the average annual income of individuals in the service area.

Three forecasting methods will be presented and compared in this chapter. these are two nonlinear Neural Network (NN) forecasting denoted as NN-1, and NN-2 and forecasting models used by the utilities studied. The models used by the utilities are in general, linear regression models. Forecasts using the NN-1 neural networks models are obtained by setting the stopping criterion in the Monte Carlo filtering process to be that of equation 3.13. Using this stopping criterion, each network search (stochastic filtering) is terminated at a preset training error level. The networks with acceptable performance on the validation set are selected in each model structure. Different model structures are ranked based on the lowest mean and variance of the selected models in the model structure. However, to obtain an adequate number of networks with good performance on the validation set in each model structure, the complete Monte Carlo process is repeated several times with different stopping error levels. The selected models and model structures for different training error levels are not similar. The error level which results in the best performance on the validation set is selected for final forecasting. The best selected model structures are used to forecast the energy sales in the backcasting period. In this chapter the forecasts using this method are denoted as NN-1 forecasts.

Forecasts using the NN-2 neural networks models are obtained by setting the stopping criterion in the Monte Carlo filtering process to be that of equation 3.14. Using this stopping criterion the training set and the validation set errors are computed at each iteration. If the training set and the validation set errors are equal, the network parameters are stored. To avoid neglecting any possible second crossing between the training and the validation set error profiles, the training process is continued for a large number of iterations. The networks within a model structure which satisfy the above condition are ranked based on the lowest crossing point error levels. Similar procedures are followed for different model structures which are chosen in the deterministic part of the Monte Carlo filtering process. Finally, different model structures are ranked based on the lowest mean and variance of the crossing point errors of models in each model structure. The best selected models are used to forecast the energy sales in the backcasting period. In this chapter the forecasts using this method are denoted as NN-2 forecasts.

IV.3 Texas Municipal Power Agency (TMPA)

TMPA is an organization of four municipally owned electric utilities in the state of Texas. The member cities include City of Bryan, City of Denton, City of Garland and City of Greenville. In the fiscal year (FY) 1993 the total energy sales for the TMPA member utilities was 3,789,727 *MWh* and the peak load demand was 808 *MW*. The breakdown of energy sales and peak load demand among the member cities for FY 1993 is shown in Table 4.1.

Forecast modeling of the City of Bryan will be presented in the next section.

IV.4 City of Bryan Forecasting Models

The City of Bryan system is divided into a city system and a rural system. The customers in the city system include a residential class, a small, a medium and a large commercial class, a school class and an interdepartmental class. For the purpose of studies reported here, the three commercial classes have been combined into a single city commercial class because of the sparsity of available data. The contribution of the school class and of the interdepartmental class to total system sales is less than 5%, therefore no sales forecasting models were developed for these two classes. The customers of the rural system of the City of Bryan are classified into a residential class and a commercial class. Thus, residential class and commercial class forecasting models were developed for the city and rural divisions of the City of Bryan. In addition a peak load demand model was developed for the entire City of Bryan system.

Table 4.1: Texas Municipal Power Agency Member Utilities Sales and Peak Load Demand (for FY 1993).

City	Energy Sales (% of Total)	Peak Load Demand (% of Total)
Bryan	20.7	21
Denton	21	22
Garland	47.3	47.1
Greenville	11	9.9
Total	3,787,727 MWh	808 MW

IV.4.1 City Residential Sales Model

Four explanatory (exogenous) input variables were selected to model and forecast the sales to the city residential customers. These are year, the number of customers in the service area, annual CDDs, and the adjusted price of electricity.

Figure 4.1 is a graph of the time series of historical and forecasted number of city residential customers. Table 4.2 shows the historical number of city residential customers from fiscal year 1970 to 1990 as well as the number of residential customers forecasted by the City of Bryan from fiscal year 1984 to 1990. The data for the forecasted number of residential customers were obtained from the City of Bryan 1984 System Load Forecast [13]. Despite the steady increase in the number of customers during the period 1970 to 1983, the number of residential customers remained almost constant during the period of 1984 to 1990. Therefore, the City of Bryan's forecast of

the number of city residential customers during the 1984 to 1990 period were in substantial error.

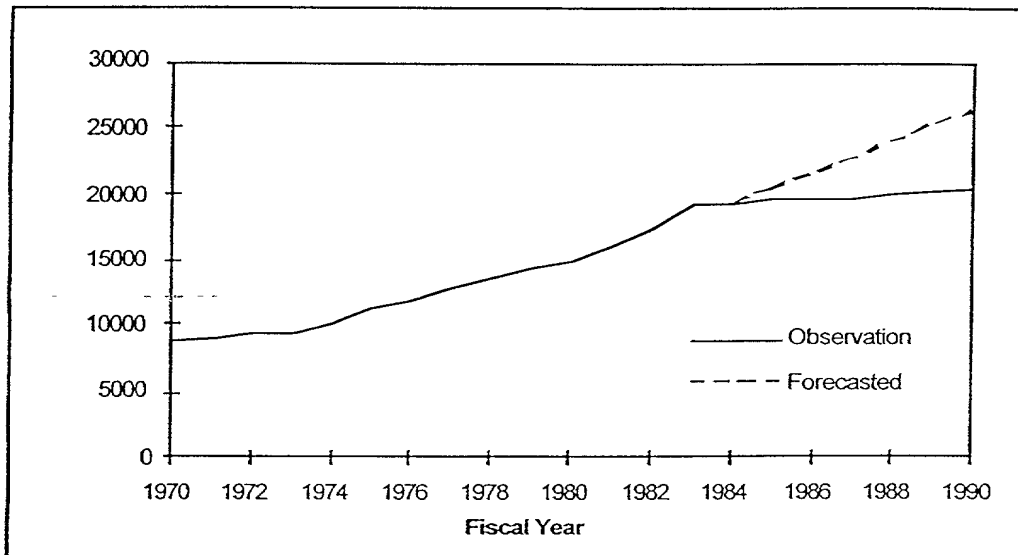


Figure 4.1: City Residential Customers Time Series for the City of Bryan.

Figures 4.2 and 4.3 and Table 4.3 present the historical and the forecasted annual CDDs time series. In the City of Bryan 1984 System Load Forecast [13] the annual CDDs are based on monthly CDDs, while in the City of Bryan 1988 System Load Forecast, the annual CDDs are based on the daily CDDs. The CDDs calculated from the two methods have similar trends in the period 1970 to 1990, but they exhibit different average values. In this study, CDDs from the City of Bryan 1988 System Load Forecast [13] have been used in developing the City of Bryan neural networks models. Table 4.3 lists the historical annual CDDs from fiscal year 1970 to 1990, as well as the forecasted annual CDDs from fiscal year 1984 to 1990. For forecasting purposes, the annual CDDs in the period 1984 to 1990 are assumed constant and are set equal to the average CDDs of the previous 14 years.

[illegible]

Table 4.2: City Residential Customers Time Series for the City of Bryan.

Fiscal Year	Customers	Forecasted Customer ^a	Error (%)
1970	8876		
1971	9066		
1972	9378		
1973	9570		
1974	10138		
1975	11242		
1976	11790		
1977	12672		
1978	13606		
1979			
1980	14942		
1981	15985		
1982	17470		
1983	19340		
1984	19381	19371	-0.1
1985	19658	20492	4.2
1986	19577	21636	10.5
1987	19567	22801	16.5
1988	19979	23988	20.1
1989	20214	25197	24.7
1990	20430	26430	29.4

^a Used in conditional forecasting

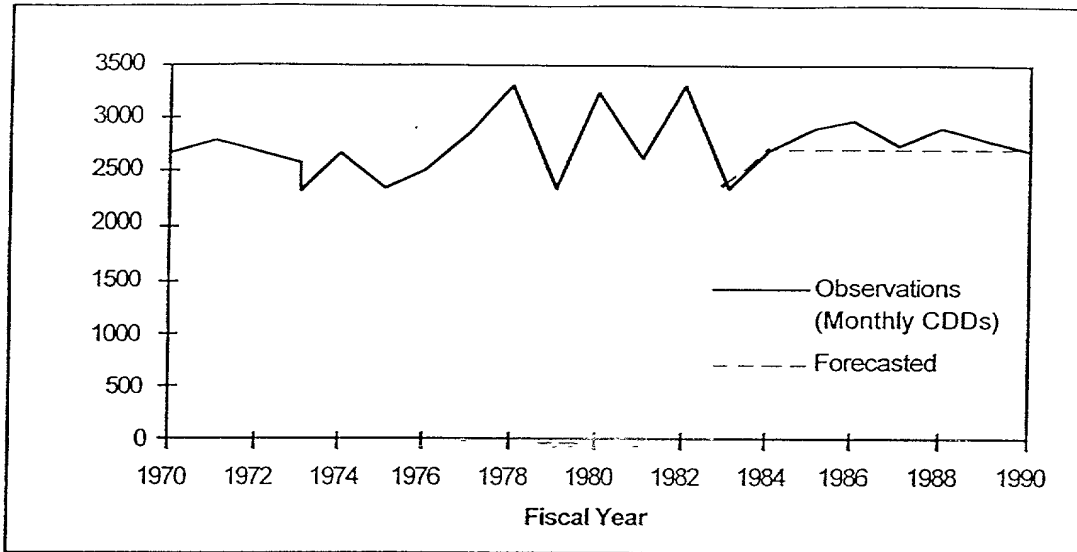


Figure 4.2: Cooling Degree Days (CDDs) Time Series for the City of Bryan (Monthly Calculation).

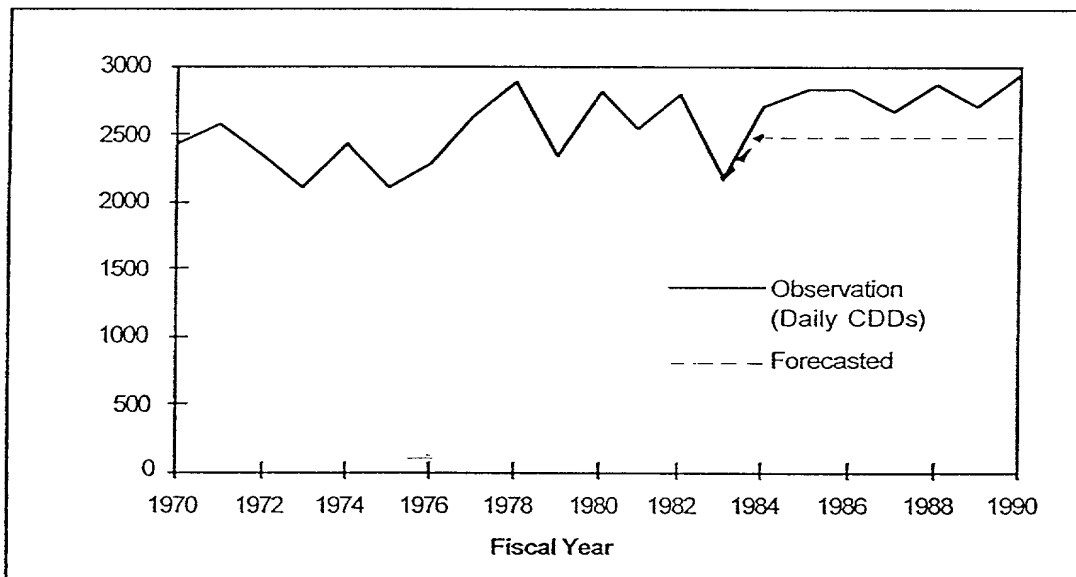


Figure 4.3: Cooling Degree Days (CDDs) Time Series for the City of Bryan (Daily Calculation).

Table 4.3: Cooling Degree Days (CDDs) Time Series for the City of Bryan.

Fiscal Year	Monthly Calculations	Forecasted CDD ^a	Error (%)	Daily Calculation	Forecasted CDD ^b	Error (%)
1970	2654			2427		
1971	2788			2561		
1972	2580			2353		
1973	2321			2094		
1974	2658			2431		
1975	2329			2102		
1976	2511			2284		
1977	2877			2633		
1978	3300			2886		
1979	2324			2357		
1980	3233			2817		
1981	2636			2550		
1982	3289			2805		
1983	2356			2159		
1984	2703	2704	0.0	2696	2461	-8.7
1985	2894	2704	-6.6	2841	2461	13.4
1986	2982	2704	-9.3	2837	2461	-13.2
1987	2747	2704	-1.6	2647	2461	-7.0
1988	2888	2704	-6.4	2861	2461	-14
1989	2792	2704	-3.2	2695	2461	-8.7
1990	2704	2704	0.0	2944	2461	-16.4

^a Used in utility's conditional forecasting^b Used in NN conditional forecasting

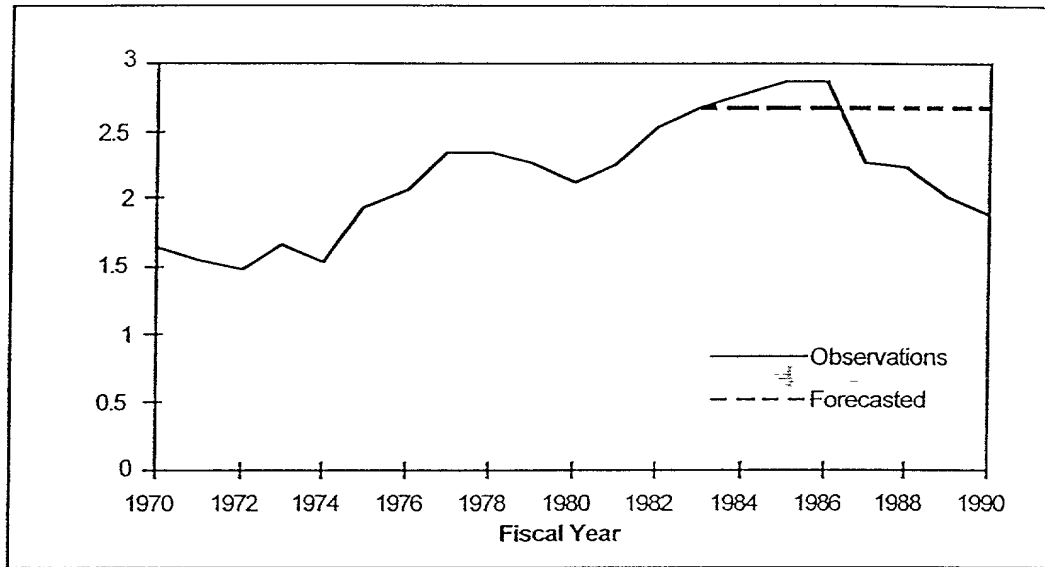


Figure 4.4: City Residential Adjusted Price of Electricity Time Series for the City of Bryan.

forecasted adjusted price of electricity from 1984 to 1990. The drop in the adjusted price of electricity during the period 1985 to 1990 is mainly due to the increase in the Consumer Price Index (CPI) while the actual price of electricity remained constant.

To model and forecast the energy sales in the city residential class using NN-1 and NN-2 nonlinear forecasting methods, the aforementioned four time series, as well as the sales in MWh were preprocessed. The preprocessing parameters were selected to ensure that the historical values and the forecasted values of the exogenous and endogenous variables fall within the linear region of the nonlinear activation functions.

Table 4.4: City Residential Adjusted Price of Electricity Time Series for the City of Bryan.

Fiscal Year	Adjusted Price (c/kwh)	Forecasted Adjusted Price (c/kwh) ^a	Error (%)
1970	1.63		
1971	1.54		
1972	1.47		
1973	1.65		
1974	1.53		
1975	1.93		
1976	2.09		
1977	2.32		
1978	2.32		
1979	2.26		
1980	2.12		
1981	2.25		
1982	2.53		
1983	2.65		
1984	2.775	2.65	-4.5
1985	2.874	2.65	-7.8
1986	2.853	2.65	-7.1
1987	2.277	2.65	16.4
1988	2.234	2.65	18.6
1989	1.992	2.65	33.0
1990	1.872	2.65	41.6

^a Used in conditional forecasting

Tables 4.5 and 4.6 show the preprocessing parameters and the results of the Monte Carlo filtering process using NN-1 and NN-2 forecasting methods. The historical data for the period 1970 to 1980 was used as the training set and the historical data for the period 1981 through 1983 was used as the validation set. The models developed based on 14 years of historical data from 1970 through 1983 were used to forecast the energy sales for a period of 7 years, from 1984 through 1990.

Using the NN-1 forecasting method, the Monte Carlo filtering process selected model structures with one past output and with 2, 3, and 4 hidden nodes respectively. Initial seed numbers of the top performing models of each model structure are shown in Table 4.5. Using the NN-2 forecasting method, the Monte Carlo filtering process selected model structures with one past output with 2 hidden nodes, one past output with 3 hidden nodes, and 2 past output with 3 hidden nodes. Initial seed numbers of top performing models of each model structure are shown in Table 4.6.

From Tables 4.5 and 4.6, it is clear that different stopping criteria resulted in the selection of different model structures. Using the NN-2 forecasting method, the average error and variance of the top performing models for each of the best 3 model structures are (0.035, 0.008), (0.039, 0.01) and (0.07, 0.013), respectively. It is clear that using the best 2 model structures, *i.e.* one past output, and 2 or 3 hidden nodes resulted in the selection of a number of forecasting models with less than 4% error on both the training and the validation data sets. However, a model structure with 2 past outputs and 3 hidden nodes resulted in the selection of forecasting models which on the average had 7% error on the training and the validation data sets.

The above discussion shows that different stopping criteria result in selection of different models. Using NN-1, the best generalization capability was obtained when the training process was terminated as 3.5% error. Using NN-2, the average crossing point

error levels of the first two model structures were 3.5% and 3.9%, which are close to that of NN-1. Two stopping criteria resulted in terminating the training process at similar

Table 4.5: City of Bryan City Residential NN-1 Forecasting Model Specifications.

Data Transformations			
Time	$\frac{Year - 1969}{21}$		
Customers	$\frac{Customers - 28000}{28000}$		
CDD	$\frac{CDD - 1500}{2000}$		
Adjusted Price	$\frac{Adjusted Price - 1}{2}$		
Sales	$\frac{Sales}{310000}$		
Training Parameters			
Learning Rate	0.01		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Training Error	3.5%		
Max. Number of Iterations Allowed	10000		
Forecasting Model Structures			
Number of Past Outputs	1	1	1
Number of Hidden Nodes	2	3	4
Top Performing NN-1 Models	Seed Numbers		
1	195174	5334	146630
2	199442	421910	267270
3	219334	70966	160962
4	78822	163542	286774
5	210498	122422	491494
6	109906	90230	252486
7	247874	33638	342198
8	140130	142966	46646
9	195906	390950	22930
10	152882	45574	157110

Table 4.6: City of Bryan City Residential NN-2 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	21		
Customers	<u>Customers</u>		
	28000		
CDD	<u>CDD - 1500</u>		
	2000		
Adjusted Price	<u>Adjusted Price - 1</u>		
	2		
Sales	<u>Sales</u>		
	310000		
Training Parameters			
Learning Rate	0.1		
Momentum Coefficient	0.3		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Max. Number of Iterations Allowed	10000		
Forecasting Model Structures			
Number of Past Outputs	1	1	2
Number of Hidden Nodes	2	3	3
Top Performing NN-2 Models	Seed Numbers		
1	460245	263661	120727
2	419967	40901	216413
3	42285	107093	71855
4	269231	490013	378613
5	356397	53525	298751
6	146407	62293	459287
7	236255	299597	276327
8	189877	267669	99999
9	472303	141261	435511
10	442991	42429	158357

error levels.

In 1984 the City of Bryan developed a forecast model for energy sales to its city residential customers [13]. This model is a linear regression model and is given in equation (4.1). The primary contributors to energy consumption in the city residential class are the number of customers, CDDs, and the price of electricity. In addition to these explanatory variables, a “crisis” variable is used to explain major discontinuities in the historical data which could not be explained by the main variables. This variable has a value of 1 from 1975 through 1977 and 0 for all other years.

$$\begin{aligned} \text{Sales} = & -48136.773 + 12.8822704 (\text{No. of Customers}) \\ & + 18.9801553 (\text{Cooling Degree Days}) - 20777.677 (\text{Adjusted Price}) \\ & - 2747.6719 (\text{Crisis}) \end{aligned} \quad (4.1)$$

Figures 4.5 and 4.6 show the unconditional forecast and forecast errors of energy sales in the city residential class. The historical energy sales and numerical values of unconditional forecasts and forecast errors are also shown in Table 4.7. In developing the unconditional neural network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The average errors of the forecast prepared by the City of Bryan and of neural network forecasts NN-1 and NN-2 are 4.4%, 3.6% and 3.6%, respectively. The City of Bryan forecast underestimated the sales in 1984 by 9.1%, while NN-1 and NN-2 underestimated the sales in 1984 by 5.2% and 7.7%, respectively. The maximum error of the City of Bryan forecast is 9.1% in 1984. The maximum annual forecast error for NN-1 is 6.3% in 1986 and for NN-2 is 7.7% in 1984.

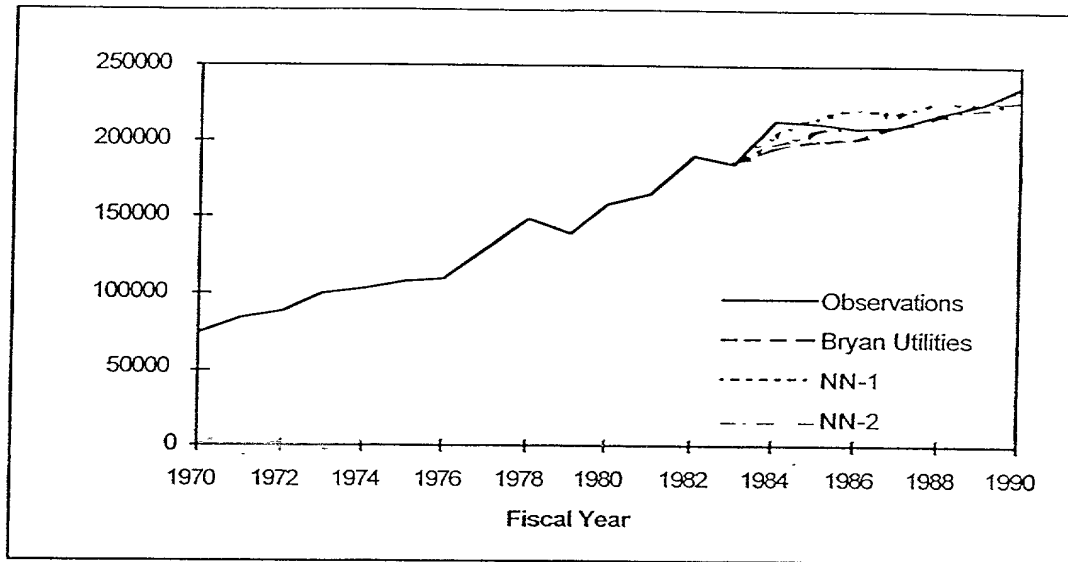


Figure 4.5: Unconditional Forecast of the City Residential Energy Sales for the City of Bryan.

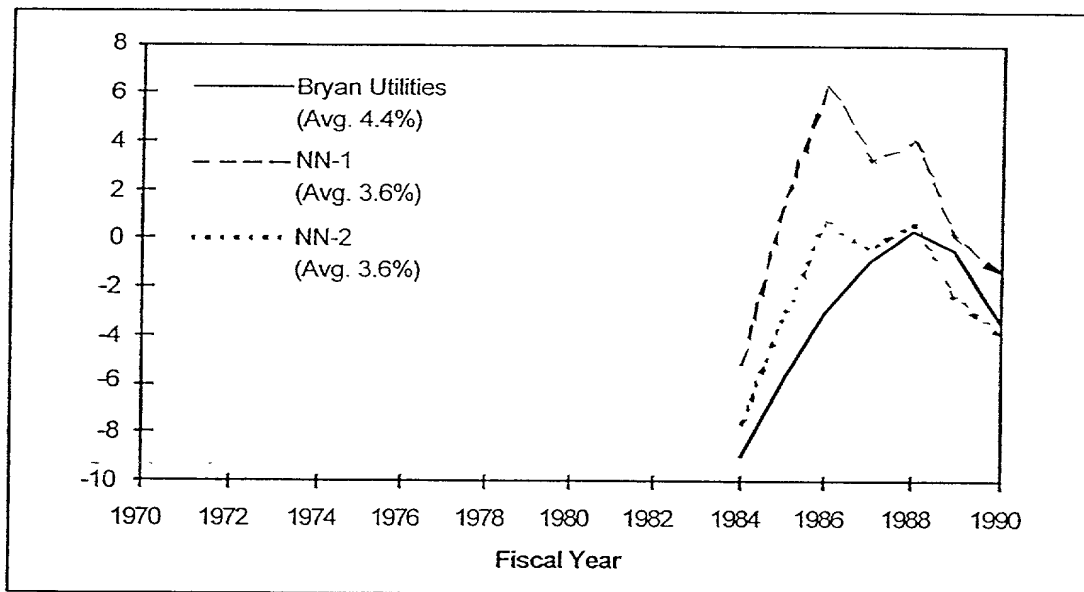


Figure 4.6: Unconditional Forecast Errors of the City Residential Energy Sales for the City of Bryan.

Table 4.7: Unconditional Energy Sales Forecasts and Forecasting Errors of the City Residential Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Unconditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	74030						
1971	82958						
1972	89745						
1973	99952						
1974	103068						
1975	107629						
1976	109980						
1977	127961						
1978	149597						
1979	138895						
1980	159641						
1981	165912						
1982	191520	Unconditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	185953	Sales	Error	Sales	Error	Sales	Error
1984	214618	195183	-9.1	203564	-5.2	198036	-7.7
1985	212105	200329	-5.6	215747	1.7	205393	-3.2
1986	207539	201379	-3	220717	6.3	209128	0.8
1987	210360	208764	-0.8	217085	3.2	209482	-0.4
1988	216769	217637	0.4	225388	4	217986	0.6
1989	224960	223882	-0.5	225102	0.1	219799	-2.3
1990	235767	227474	-3.5	232550	-1.4	226686	-3.9
Avg.			4.4		3.6		3.6

^a (MWh)

Figures 4.7 and 4.8 show the conditional forecast and forecast errors of energy sales in the city residential class. The historical energy sales and numerical values of conditional forecasts and forecast errors are also shown in Table 4.8. In developing the conditional Neural Network forecasts, the information for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The projections of exogenous variables used in the City of Bryan 1984 System Load forecast [13] were used to generate conditional forecasts. While all forecasts overestimated energy sales, the average error for the City of Bryan 1984 System Load forecast was 15.9%, compared to 8.6% and 5.9% for NN-1 and NN-2, respectively. The City of Bryan 1984 System Load Forecast overestimated the 1990 energy sales by 22.4%, while the NN-1 and NN-2 overestimated the sales in 1990 by 9.6% and 4.9%, respectively. The maximum annual conditional forecast error for the City of Bryan 1984 System Load Forecast is 22.4% in 1990. The maximum annual conditional forecast error for NN-1 is 10.8% in 1989 and for NN-2 is 9.4% in 1984. As expected, conditional forecasts result in an increase in the annual forecasting errors for both the NN models and the City of Bryan load forecast since the error in the forecast of the exogenous variables also contributes to the total forecasting error. However, the neural networks are seen to be substantially more accurate and tolerant of exogenous variable forecast errors than the linear regression model used by the City of Bryan.

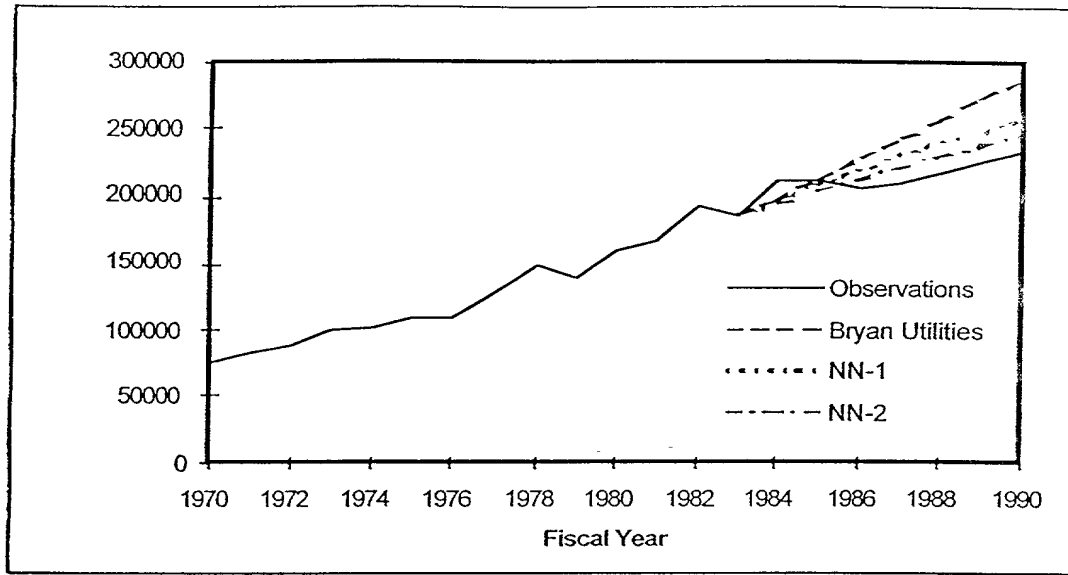


Figure 4.7: Conditional Forecast of the City Residential Energy Sales for the City of Bryan.

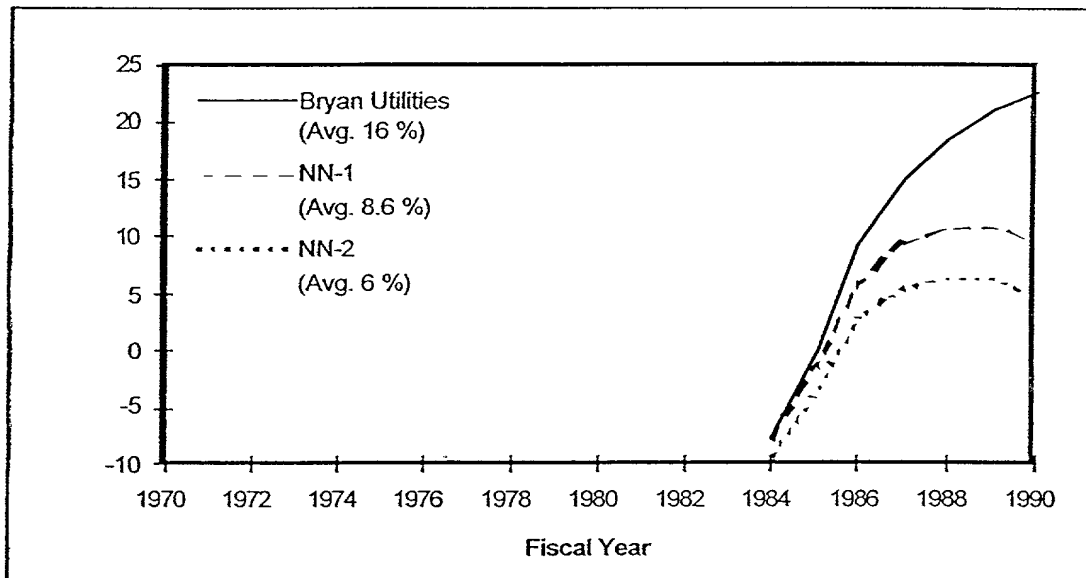


Figure 4.8: Conditional Forecast Errors of the City Residential Energy Sales for the City of Bryan.

Table 4.8: Conditional Energy Sales Forecasts and Forecasting Errors of the City Residential Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Conditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	74030						
1971	82958						
1972	89745						
1973	99952						
1974	103068						
1975	107629						
1976	109980						
1977	127961						
1978	149597						
1979	138895						
1980	159641						
1981	165912						
1982	191520						
		Conditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	185953	Sales	Error	Sales	Error	Sales	Error
1984	214618	197662	-7.9	197502	-8	194538	-9.4
1985	212105	212114	0.0	208634	-1.6	203570	-4.0
1986	207539	226842	9.3	219435	5.7	212792	2.5
1987	210360	241850	15	229857	9.3	221710	5.4
1988	216769	257143	18.6	239848	10.6	230534	6.4
1989	224960	272723	21.2	249356	10.8	239085	6.3
1990	235767	288597	22.4	258353	9.6	247433	4.9
Avg.			15.9		8.6		5.9

^a (MWh)

IV.4.2 City Commercial Sales Model

Three explanatory (exogenous) input variables were selected to model and forecast the sales to the city commercial customers. These are year, the number of customers in the service area and the adjusted price of electricity.

Figure 4.9 is a graph of the time series of historical and forecasted number of city commercial customers time series. Table 4.9 shows the historical city commercial customers from fiscal year 1970 to 1990 as well as the number of commercial customers forecasted by the City of Bryan from fiscal year 1984 to 1990. The data for the forecasted commercial customers were obtained from the City of Bryan 1984 System Load Forecast [13]. Despite the steady increase in the number of customers during the period 1970 to 1982, the number of commercial customers remained almost constant during the forecasting period of 1984 to 1990. This accounts for the large error in the number of forecasted customers by the City of Bryan.

Figure 4.10 shows the historical and the forecasted adjusted price of electricity for the city commercial customers in cents per KWh. Table 4.10 lists the historical adjusted price of electricity from fiscal year 1970 to 1990 as well as the forecasted adjusted price of electricity from 1984 to 1990. The drop in the adjusted price of electricity for the period 1985 to 1990 was mainly due to the increase in the Consumer Price Index (CPI) while the actual price of electricity remained constant.

To model and forecast the energy sales in the city commercial class using NN-1 and NN-2 nonlinear forecasting techniques, the aforementioned three time series, as well as the sales in MWh were preprocessed. The preprocessing parameters were selected to ensure that the historical values and the forecasted values of the exogenous and endogenous variables fall within the linear region of the nonlinear activation functions.

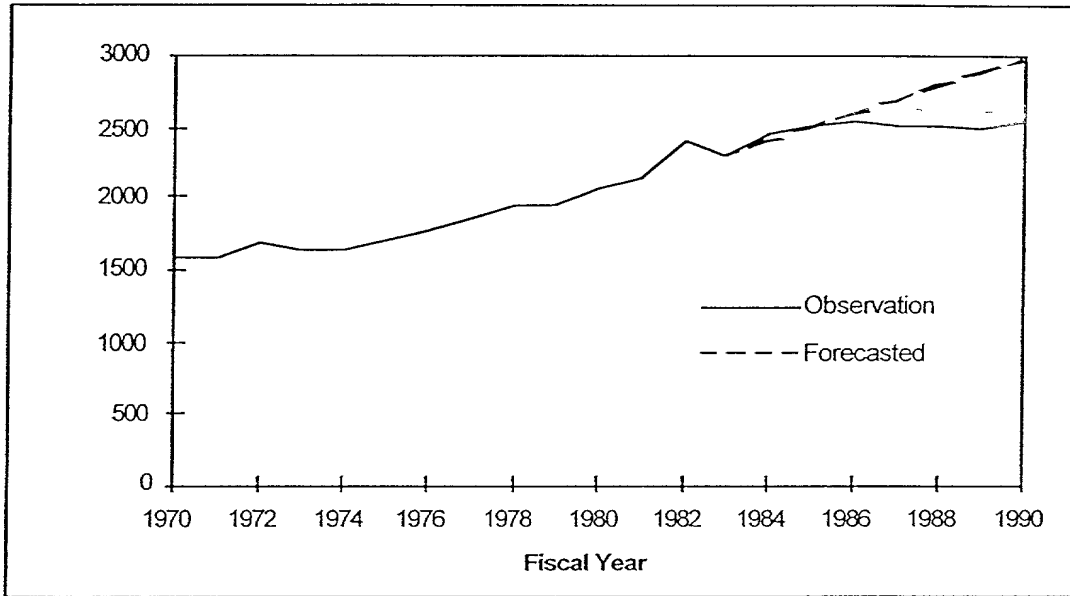


Figure 4.9: City Commercial Customers Time Series for the City of Bryan.

Table 4.9: City Commercial Customers Time Series for the City of Bryan.

Fiscal Year	Customers	Forecasted Customer ^a	Error (%)
1970	1575		
1971	1583		
1972	1689		
1973	1645		
1974	1651		
1975	1715		
1976	1782		
1977	1864		
1978	1948		
1979	1957		
1980	2076		
1981	2147		
1982	2396		
1983	2305		
1984	2439	2396	-1.76
1985	2516	2495	-0.83
1986	2546	2592	1.81
1987	2510	2689	7.13
1988	2512	2784	10.83
1989	2500	2878	15.12
1990	2527	2971	17.57

^a Used in conditional forecasting

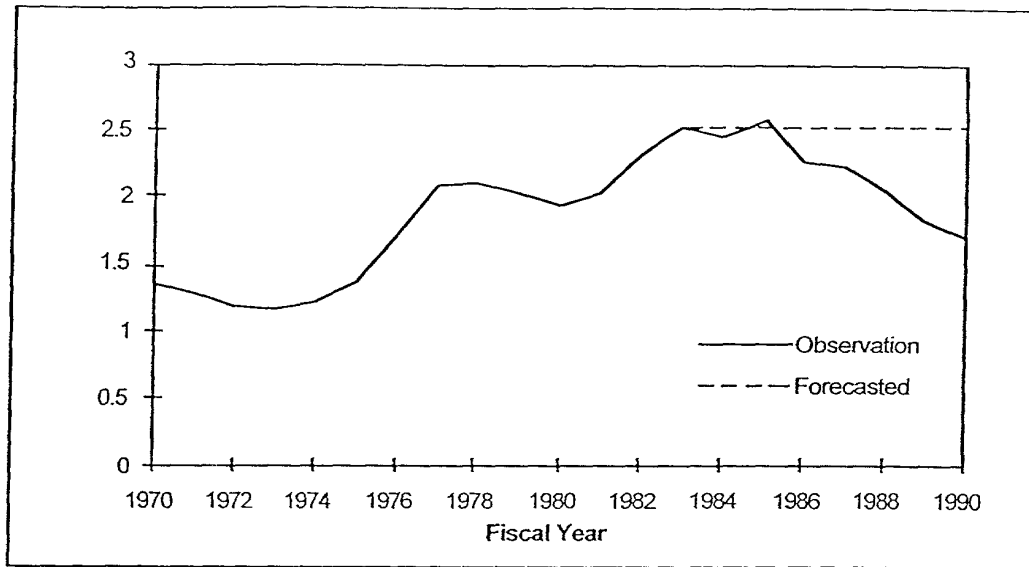


Figure 4.10: City Commercial Adjusted Price of Electricity Time Series for the City of Bryan.

Tables 4.11 and 4.12 show the preprocessing parameters and results of the Monte Carlo filtering process using NN-1 and NN-2 forecasting methods. The historical data for period 1970 to 1980 was used as the training set and the historical data for period 1981 through 1983 was used as the validation set. The models developed based on 14 years of historical data were used to forecast the energy sales for a period of 7 years, from 1984 through 1990.

Using NN-1 forecasting method, the Monte Carlo filtering process selected model structures with one past output and with 2, 3 and 4 hidden nodes respectively. Initial seed numbers of top performing models of each model structure are shown in Table 4.11. Using NN-2 forecasting method, the Monte Carlo filtering process selected model structures with 1 past output with 3 hidden nodes, 2 past outputs with 3 hidden nodes and 1 past output with 2 hidden nodes. Initial seed numbers of top performing models of each model structure are shown in Table 4.12.

Table 4.10: City Commercial Adjusted Price of Electricity Time Series for the City of Bryan.

Fiscal Year	Adjusted Price (c/kwh)	Forecasted Adjusted Price (c/kwh) ^a	Error (%)
1970	1.34		
1971	1.29		
1972	1.20		
1973	1.18		
1974	1.22		
1975	1.38		
1976	1.72		
1977	2.09		
1978	2.10		
1979	2.04		
1980	1.95		
1981	2.04		
1982	2.34		
1983	2.52		
1984	2.45	2.52	2.9
1985	2.61	2.52	-3.5
1986	2.28	2.52	10.5
1987	2.24	2.52	12.5
1988	2.08	2.52	21.2
1989	1.84	2.52	37.0
1990	1.70	2.52	48.2

^a Used in conditional forecasting

Table 4.11: City of Bryan City Commercial NN-1 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	25		
Customers	<u>Customers</u>		
	3200		
Adjusted Price	<u>Adjusted Price - 1</u>		
	2.5		
Sales	<u>Sales</u>		
	320000		
Training Parameters			
Learning Rate	0.003		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Training Error	7%		
Max. Number of Iterations Allowed	5000		
Forecasting Model Structures			
Number of Past Outputs	1	1	1
Number of Hidden Nodes	2	3	4
Top Performing NN-1 Models	Seed Numbers		
1	423831	453935	506127
2	275031	511629	127949
3	156103	133655	249327
4	423813	511517	478045
5	218325	210061	318437
6	466925	324295	118437
7	292503	522911	249581
8	31495	269311	400383
9	393911	218159	3369498
10	517917	82095	422103

Table 4.12: City of Bryan City Commercial NN-2 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	25		
Customers	<u>Customers</u>		
	28000		
CDD	<u>CDD - 1500</u>		
	2000		
Adjusted Price	<u>Adjusted Price - 1</u>		
	2		
Sales	<u>Sales</u>		
	310000		
Training Parameters			
Learning Rate	0.1		
Momentum Coefficient	0.3		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Max. Number of Iterations Allowed	10000		
Forecasting Model Structures			
Number of Past Outputs	1	2	1
Number of Hidden Nodes	3	3	2
Top Performing NN-2 Models	Seed Numbers		
1	405391	513853	411167
2	303941	436317	511719
3	454941	41157	5807
4	123981	292471	269519
5	297725	38103	256303
6	262751	154623	212677
7	360287	44407	4383
8	395703	196695	328597
9	504007	519863	302301
10	333565	252981	99269

From Tables 4.11 and 4.12 it is clear that different stopping criteria resulted in the selection of different model structures. Using NN-2 forecasting method, the average error and variance of the top performing models for the best 3 model structures are (0.062, 0.007), (0.072, 0.006) and (0.064, 0.009), respectively. It is clear that using the NN-2 technique, the crossing between the training and the testing set error profiles occurs at about 6% error levels. It is interesting to note that using NN-1 technique, the best generalization capability was obtained when the training was terminated at 7% error level. While the error level was set manually in NN-1 technique, the estimation process derived a value close to this error level in NN-2 forecasting method.

In 1984 the City of Bryan developed a forecast model for energy sales to its city commercial customers [13]. This model is a linear regression model and is given in equation (4.2). The primary contributors to energy consumption in the city commercial class are the number of customers and the price of electricity. In addition to these explanatory variables, a “crisis” variable is used to explain major discontinuities in the historical data which could not be explained by the main variables. This variable has a value of 1 in the year 1982, and 0 for all other years.

$$\begin{aligned} \text{Sales} = & -32403.028 + 92.8203773 (\text{No. of Customers}) \\ & + 3950.5666 (\text{Adjusted Price}) - 2747.6719 (\text{Crisis}) \end{aligned} \quad (4.2)$$

Figures 4.11 and 4.12 show the unconditional forecast and forecast errors of energy sales in the city commercial class. The historical energy sales and numerical values of unconditional forecasts and forecast errors are also shown in Table 4.13. In developing the unconditional neural network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The average errors of the forecast prepared by the City of Bryan and neural network forecasts NN-1 and NN-2 are 15.7%, 14.1% and 10.5%, respectively.

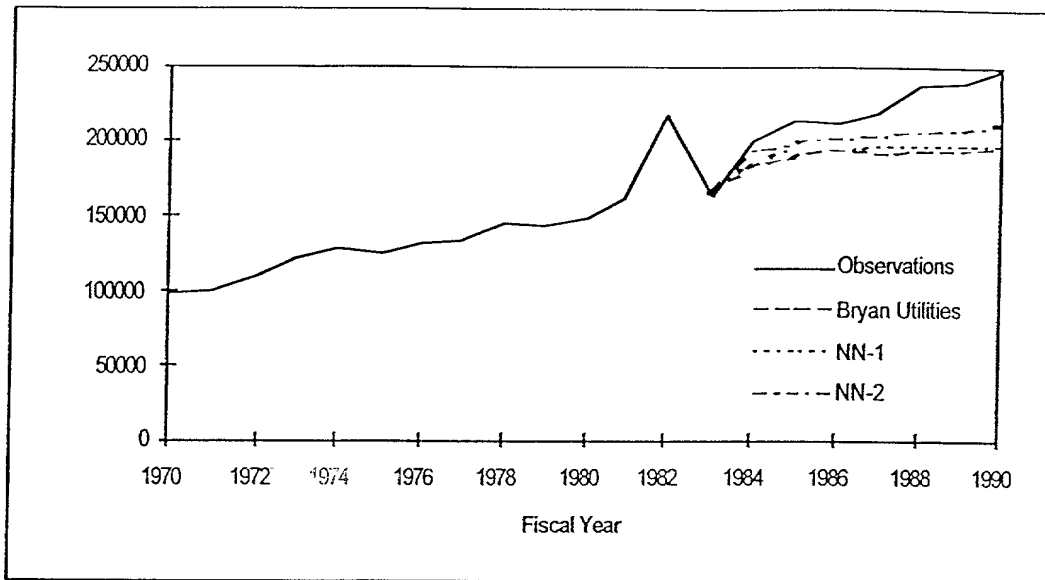


Figure 4.11: Unconditional Forecast of the City Commercial Energy Sales for the City of Bryan.

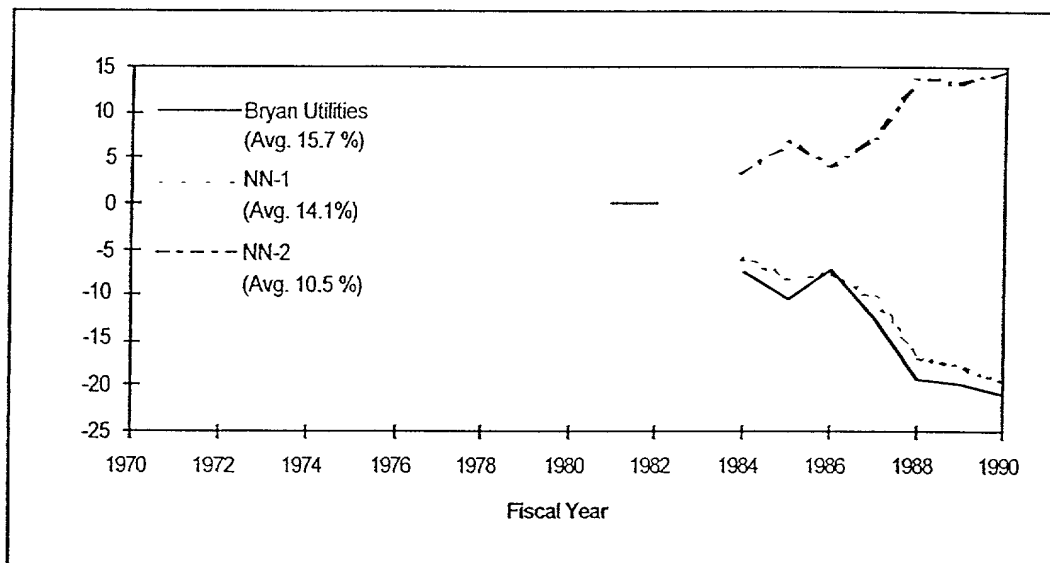


Figure 4.12: Unconditional Forecast Errors of the City Commercial Energy Sales for the City of Bryan.

Table 4.13: Unconditional Energy Sales Forecasts and Forecasting Errors of the City Commercial Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Unconditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	99120						
1971	100869						
1972	109714						
1973	122325						
1974	126752						
1975	125547						
1976	131346						
1977	133379						
1978	144955						
1979	143163						
1980	148894						
1981	162018						
1982	218587						
		Unconditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	165331	Sales	Error	Sales	Error	Sales	Error
1984	199576	184307	-7.7	187434	-6.1	193674	3.0
1985	213373	190822	-10.6	195207	-8.5	199338	6.6
1986	210553	194910	-7.4	194094	-7.8	202439	3.9
1987	219916	191727	-12.8	196993	-10.4	204150	7.2
1988	238570	192545	-19.3	198166	-16.9	206393	13.5
1989	239943	192379	-19.8	197445	-17.7	207758	13.4
1990	246334	195438	-20.7	198289	-19.5	210584	14.5
Avg.			15.7		14.1		10.5

^a (MWh)

The City of Bryan forecast underestimated the sales in 1984 by 7.7% and NN-1 underestimated the sales in 1984 by 6.1% while NN-2 overestimated the sales in 1984 by 3%. The maximum forecast error occurs for all the 3 methods in 1990. The maximum error of the City of Bryan, NN-1 and NN-2 are 20.7 %, 19.5% and 14.5% respectively.

Figures 4.13 and 4.14 show the conditional forecast and forecast errors of energy sales in the city commercial class. The historical energy sales and numerical values of conditional forecasts and forecast errors are also shown in Table 4.14. In developing the conditional Neural Network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The projections of exogenous variables used in the City of Bryan 1984 System Load forecast [13] were used to generate conditional forecasts. The average forecast error for the City of Bryan 1984 System Load forecast was 7.9% compared to 10.1% and 7.1% for NN-1 and NN-2, respectively. The City of Bryan 1984 System Load Forecast underestimated the 1990 energy sales by 5.2%. The NN-1 underestimated the 1990 energy sales by 12.2% while the NN-2 overestimated the 1990 sales by 8.6%. The maximum annual conditional forecast error for the City of Bryan 1984 System Load Forecast is 11.3% in 1985. The maximum annual conditional forecast error for NN-1 is 13.0% in 1988 and for NN-2 is 9.8% in 1988.

Evidently, the city commercial class has experienced a structural change during the 1980's. While the number of commercial customers remained almost constant, the energy sales in this class continued to increase. The unconditional forecasts, where the modeling accuracy is tested resulted in large forecast errors.

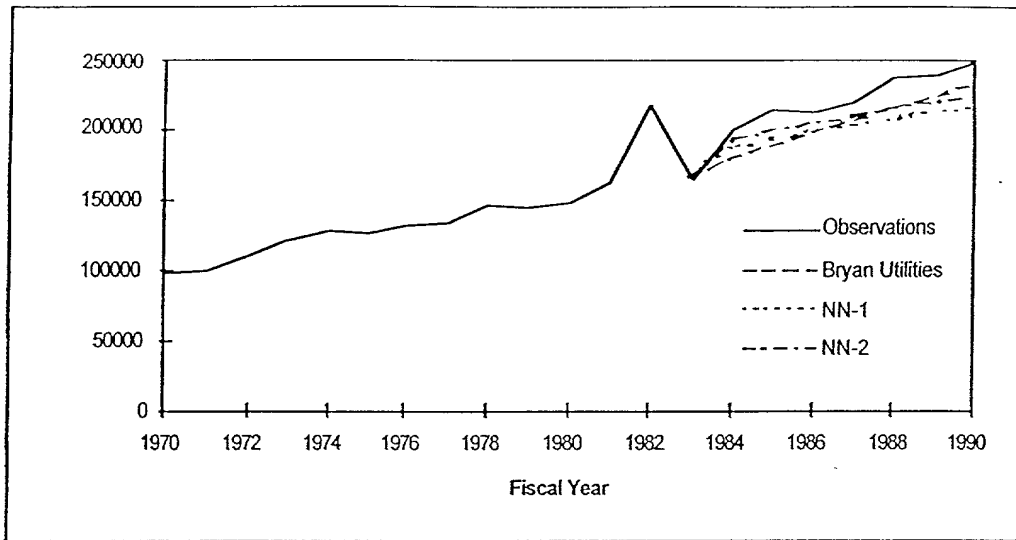


Figure 4.13: Conditional Forecast of the City Commercial Energy Sales for the City of Bryan.

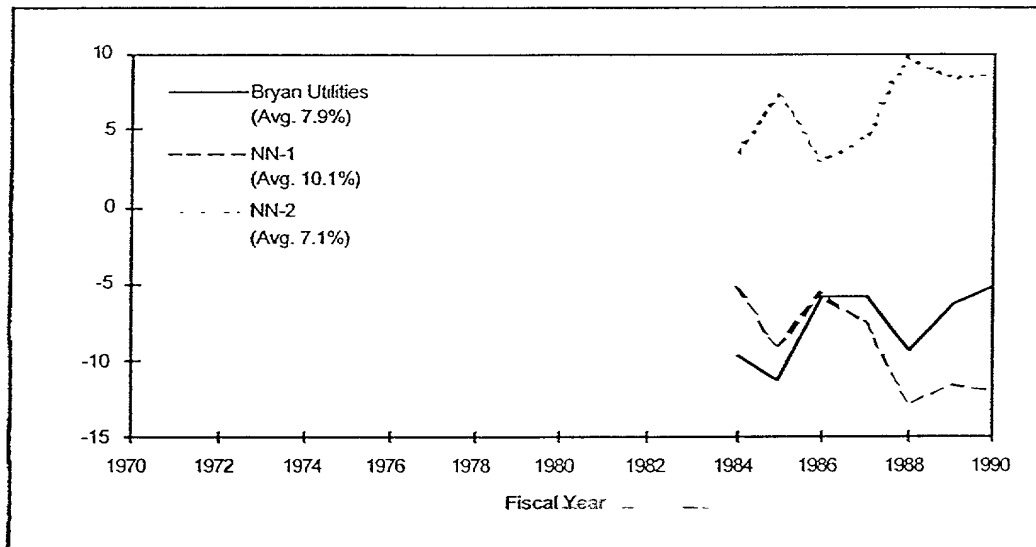


Figure 4.14: Conditional Forecast Errors of the City Commercial Energy Sales for the City of Bryan.

Table 4.14: Conditional Energy Sales Forecasts and Forecasting Errors of the City Commercial Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Conditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	99120						
1971	100869						
1972	109714						
1973	122325						
1974	126752						
1975	125547						
1976	131346						
1977	133379						
1978	144955						
1979	143163						
1980	148894						
1981	162018						
1982	218587						
		Conditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	165331	Sales	Error	Sales	Error	Sales	Error
1984	199576	180056	-9.8	188537	-5.5	192511	3.5
1985	213373	189219	-11.3	193730	-9.2	198432	7.0
1986	210553	198274	-5.8	198368	-5.8	204532	2.9
1987	219916	207224	-5.8	203019	-7.7	210029	4.5
1988	238570	216068	-9.4	207522	-13.0	215238	9.8
1989	239943	224808	-6.3	211924	-11.7	220314	8.2
1990	246334	233444	-5.2	216215	-12.2	225206	8.6
Avg.			7.9		10.1		7.1

^a (MWh)

IV.4.3 Rural Residential Sales Model

Four explanatory (exogenous) input variables were selected to model and forecast the sales to the rural residential customers. These are year, the number of customers in the service area, the annual CDDs and the adjusted price of electricity.

Figure 4.15 is a graph of the time series of historical and forecasted number of city residential customers. Table 4.15 shows the historical number of rural residential customers from fiscal year 1970 to 1990 as well as the forecasted residential customers from fiscal year 1984 to 1990. The data for the forecasted number of residential customers were obtained from the City of Bryan 1984 System Load Forecast [13]. During the year 1975 to 1982, the rural residential class experienced a steady increase in the number of customers. However, after a drop in 1983, the annual increases in the number of customers in the rural residential class have been marginal.

For the Rural Residential class, the annual CDDs used in the forecasts are the ones used for the city residential class. These are shown in Figures 4.2, 4.3 and Table 4.4.

Figure 4.16 shows the historical and the forecasted adjusted price of electricity for the rural residential customers in cents per KWh. Table 4.16 lists the historical adjusted price of electricity from fiscal year 1970 to 1990 as well as the forecasted adjusted price of electricity from 1984 to 1990. The forecasted adjusted price was assumed to be a constant 2.60 cents per *KWh*, corresponding to the adjusted price in 1983.

To model and forecast the energy sales in the rural residential class using NN-1 and NN-2 nonlinear forecasting methods, the aforementioned four time series, as well as the sales in MWh were preprocessed. The preprocessing parameters were selected to

806

ensure that the historical values and the forecasted values of the exogenous and endogenous variables fall within the linear region of the nonlinear activation functions.

111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805

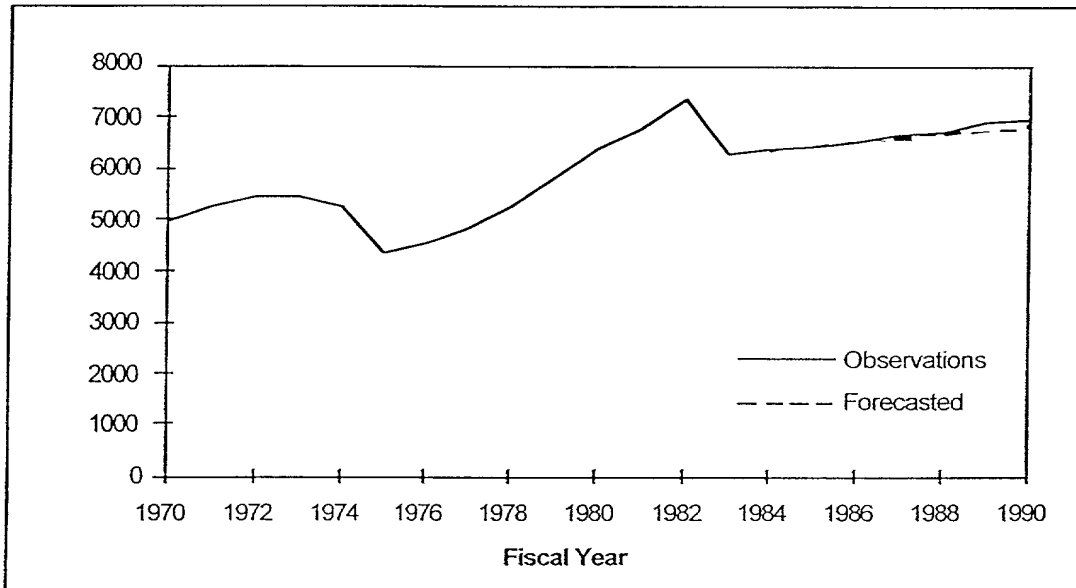


Figure 4.15: Rural Residential Customers Time Series for the City of Bryan.

Table 4.15: Rural Residential Customers Time Series for the City of Bryan.

Fiscal Year	Customers	Forecasted Customer ^a	Error (%)
1970	4976		
1971	5238		
1972	5469		
1973	5407		
1974	5228		
1975	4324		
1976	4510		
1977	4822		
1978	5248		
1979	5848		
1980	6329		
1981	6768		
1982	7323		
1983	6272		
1984	6374	6333	-0.6
1985	6399	6419	0.3
1986	6571	6502	-1.1
1987	6634	6584	-0.8
1988	6733	6665	-1.0
1989	6869	6744	-1.8
1990	6938	6822	-1.7

^a Used in conditional forecasting

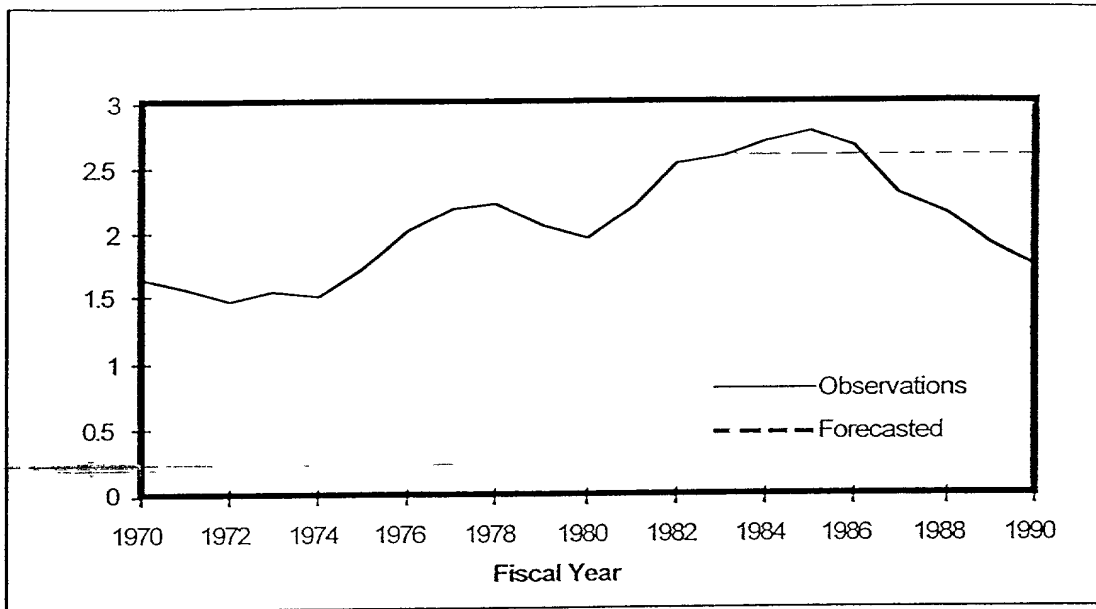


Figure 4.16: Rural Residential Adjusted Price of Electricity Time Series for the City of Bryan.

Tables 4.17 and 4.18 show the preprocessing parameters and the results of the Monte Carlo filtering process using NN-1 and NN-2 forecasting methods. The historical data for period 1970 to 1980 was used as the training set and the historical data for period 1981 through 1983 was used as the validation set. The models developed based on 14 years of historical data, were used to forecast the energy sales for a period of 7 years, from 1984 through 1990.

Using NN-1 forecasting method, the Monte Carlo filtering process selected model structures with 2 past outputs with 3 hidden nodes, 3 past outputs with 3 hidden nodes, and 4 past outputs with 4 hidden nodes. Initial seed numbers of top performing models of each model structure are shown in Table 4.17. Using NN-2 forecasting method, the Monte Carlo filtering process selected model structures with 3 past outputs with 1 hidden node, 1 past output with 1 hidden node and 2 past outputs with 1 hidden

Table 4.16: Rural Residential Adjusted Price of Electricity Time Series for the City of Bryan.

Fiscal Year	Adjusted Price (c/kwh)	Forecasted Adjusted Price (c/kwh) ^a	Error (%)
1970	1.66		
1971	1.58		
1972	1.49		
1973	1.57		
1974	1.51		
1975	1.73		
1976	2.01		
1977	2.19		
1978	2.24		
1979	2.07		
1980	1.98		
1981	2.21		
1982	2.55		
1983	2.6		
1984	2.72	2.6	-4.4
1985	2.79	2.6	-6.8
1986	2.65	2.6	-1.9
1987	2.32	2.6	12.1
1988	2.15	2.6	20.9
1989	1.93	2.6	34.7
1990	1.76	2.6	47.7

^a Used in conditional forecasting

Table 4.17: City of Bryan Rural Residential NN-1 Forecasting Model Specifications.

Data Transformations				
Time	<u>Year - 1969</u>			
	25			
Customers	<u>Customers</u>			
	28000			
CDD	<u>CDD - 1500</u>			
	2000			
Adjusted Price	<u>Adjusted Price - 1</u>			
	2			
Sales	<u>Sales</u>			
	310000			
Training Parameters				
Learning Rate	0.003			
Estimation Period	1970 - 1983			
Training Period	1970 - 1980			
Validation Period	1981 - 1983			
Forecasting Period	1984 - 1990			
Training Error	3.5%			
Max. Number of Iterations Allowed	10000			
Forecasting Model Structures				
Number of Past Outputs	2	3	3	
Number of Hidden Nodes	3	3	4	
Top Performing NN-1 Models	Seed Numbers			
1	262726	4246	144438	
2	236898	55798	274502	
3	451462	385686	449778	
4	48770	441526	247650	
5	368774	487414	472646	
6	195602	72502	193654	
7	399890	100534	517426	
8	104902	421462	42434	
9	120630	74134	73730	
10	435762	432726	384422	

Table 4.18: City of Bryan Rural Residential NN-2 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	25		
Customers	<u>Customers</u>		
	28000		
CDD	<u>CDD - 1500</u>		
	2000		
Adjusted Price	<u>Adjusted Price - 1</u>		
	2		
Sales	<u>Sales</u>		
	310000		
Training Parameters			
Learning Rate	0.1		
Momentum Coefficient	0.3		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Max. Number of Iterations Allowed	10000		
Forecasting Model Structures			
Number of Past Outputs	3	1	2
Number of Hidden Nodes	1	1	1
Top Performing NN-2 Models	Seed Numbers		
1	232483	27169	287033
2	3267	359787	520961
3	212739	257697	5465
4	418275	29827	335857
5	399971	499113	134881
6	222531	4254511	120881
7	225219	405289	452187
8	89667	333513	149537
9	25155	360401	329371
10	29891	119929	364155

node. Initial seed numbers of top performing models of each model structure are shown in Table 4.18.

From Tables 4.17 and 4.18 it is clear that different stopping criteria resulted in the selection of different model structures. Using NN-2 forecasting method, the average error and variance of the top performing models for each of the best 3 model structures are (0.034, 0.0004), (0.036, 0.0005) and (0.036, 0.0005), respectively. It is clear that top model structures have performed very similarly on the estimation set. Moreover, using NN-2 forecasting method, the crossing points between the training set and the validation set were at about 3.5% error level. Using NN-1 forecasting method, the best generalization capability was obtained when the training was terminated at 3.5% error level.

In 1984 the City of Bryan developed a forecast model for energy sales to its rural residential customers [13]. This model is a linear regression and is given in equation (4.3). The primary contributors to energy consumption in the rural residential class are the number of customers, CDDs and the price of electricity. In addition to these explanatory variables, a “crisis” variable is used to explain major discontinuities in the historical data which can not be explained by the main variables. This variable has a value of 1 prior to 1975, and 0 thereafter.

$$\begin{aligned} \text{Sales} = & -29518.316 + 14.8764926 \text{ (No. of Customers)} \\ & + 2.21708178 \text{ (Cooling Degree Days)} - 236.15972 \text{ (Adjusted Price)} \\ & - 13230.964 \text{ (Crisis)} \end{aligned} \quad (4.3)$$

Figures 4.17 and 4.18 show the unconditional forecast and forecast errors of energy sales in the rural residential class. The historical energy sales and numerical values of unconditional forecasts and forecast errors are also shown in Table 4.19. In developing

the unconditional neural network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The average errors of forecasts prepared by the City of Bryan, NN-1 and NN-2 are 12.8%, 6.5% and 3.7%, respectively. The City of Bryan forecast underestimated the sales in 1984 by 10.6%, while NN-1 overestimated the sales in 1984 by 0.3% and NN-2 underestimated the sales in 1984 by 6.2%. The maximum error of Bryan forecast is 17.5% in 1990. The maximum annual forecast error of NN-1 is 10.8% in 1990, and for NN-2 is 6.2% in 1984.

Figures 4.19 and 4.20 show the conditional forecast and forecast errors of energy sales in the rural residential class. The historical energy sales and numerical values of conditional forecasts and forecast errors are also shown in Table 4.20. In developing the conditional neural network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The projections of exogenous variables used in the City of Bryan 1984 System Load forecast [13] were used to generate conditional forecasts. While both forecasts underestimated energy sales, the average error for the City of Bryan 1984 System Load forecast was 14.4%, compared to 9% and 5.8% for NN-1 and NN-2, respectively. The City of Bryan 1984 System Load Forecast underestimated the 1984 energy sales by 11.3%, while the NN-1 and NN-2 underestimated the sales in 1984 by 2.9% and 7.2% respectively. The maximum annual conditional forecast error for the City of Bryan 1984 System Load Forecast is 20.1% in 1990. The maximum annual conditional forecast error for NN-1 is 14.5% in 1990 and for NN-2 is 10.1% in 1990. As expected, performing a conditional forecast results in an increase of the annual forecasting errors for both the neural network models and City of Bryan load forecasts, since the error in the forecast of the exogenous variables will also contribute to total forecasting error.

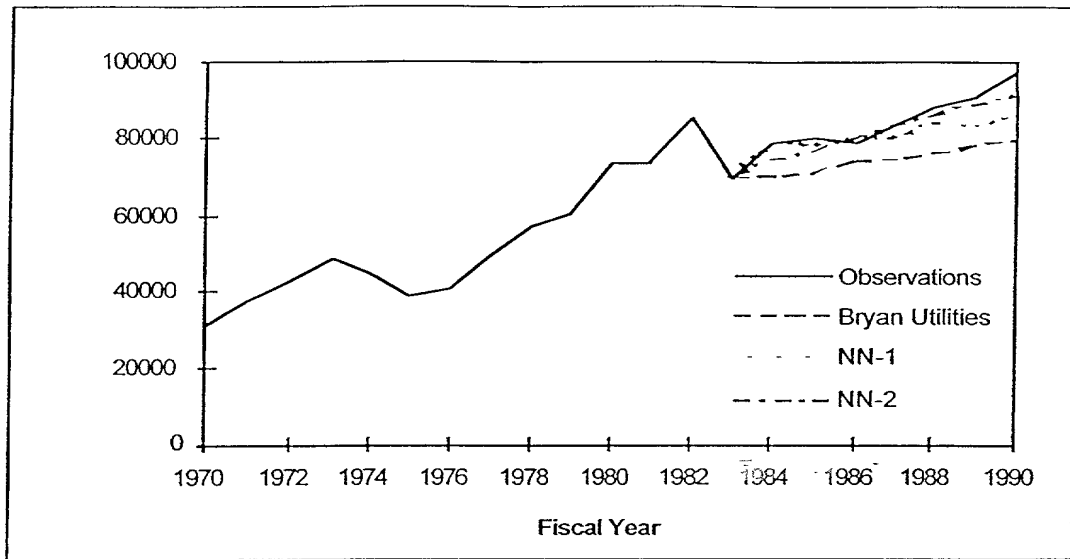


Figure 4.17: Unconditional Forecast of the Rural Residential Energy Sales for the City of Bryan.

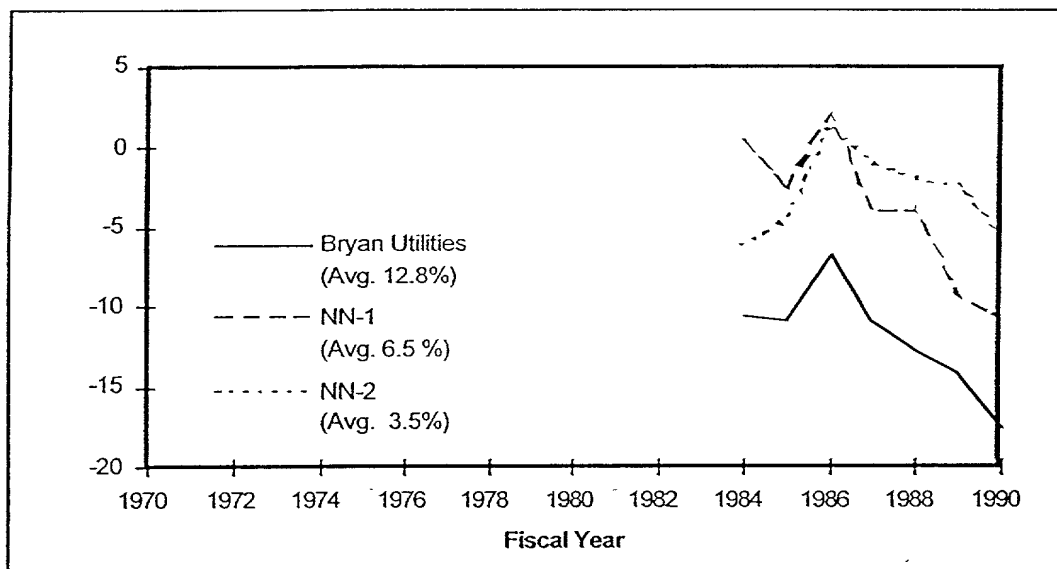


Figure 4.18: Unconditional Forecast Errors of the Rural Residential Energy Sales for the City of Bryan.

Table 4.19: Unconditional Energy Sales Forecasts and Forecasting Errors of the Rural Residential Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Unconditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	31175						
1971	37291						
1972	43000						
1973	48480						
1974	44807						
1975	39050						
1976	41278						
1977	49582						
1978	57123						
1979	60932						
1980	73200						
1981	73443						
1982	85386						
1983	70129	Unconditional Forecast					
		Bryan Utilities		NN-1		NN-2	
		Sales	Error	Sales	Error	Sales	Error
1984	78993	70639	-10.6	79264	0.3	74369	-6.2
1985	80093	71316	-11	78046	-2.6	76573	-4.6
1986	79157	73899	-6.6	80611	1.8	80155	1.2
1987	83672	74493	-11	80351	-4.0	82849	-1.0
1988	87814	76480	-12.9	84286	-4.0	86214	-1.9
1989	91063	78187	-14.1	82817	-9.1	88804	-2.5
1990	96775	79806	-17.5	86347	-10.8	91656	-5.6
Avg.			12.8		6.5		3.7

^a (MWh)

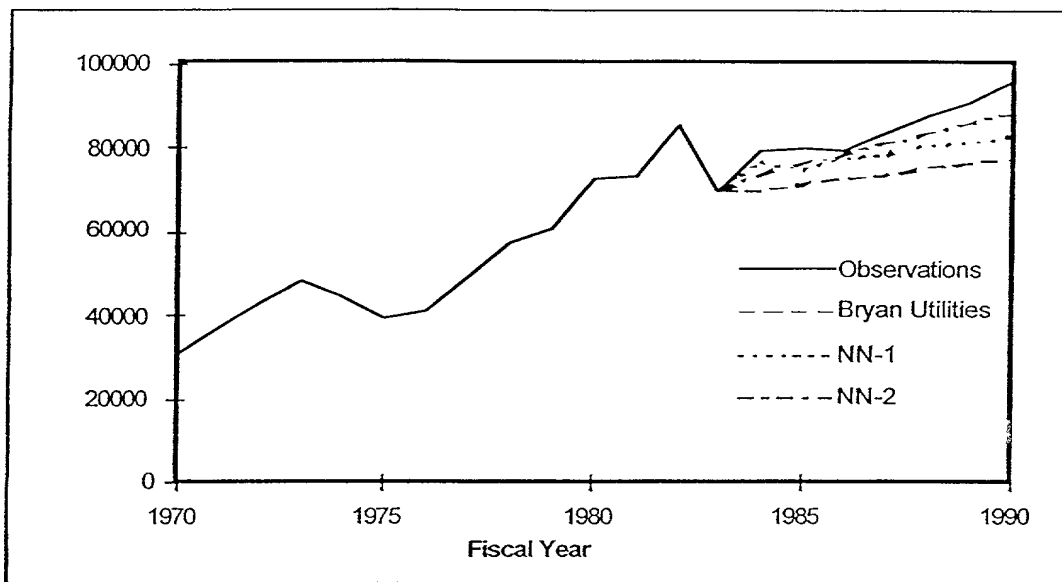


Figure 4.19: Conditional Forecast of the Rural Residential Energy Sales for the City of Bryan.

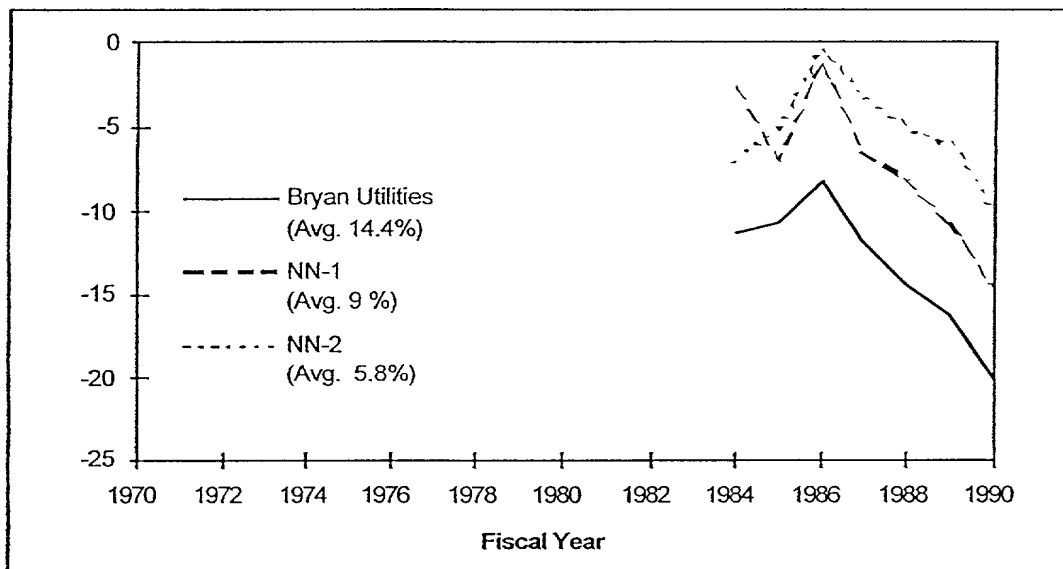


Figure 4.20: Conditional Forecast Errors of the Rural Residential Energy Sales for the City of Bryan.

Table 4.20: Conditional Energy Sales Forecasts and Forecasting Errors of the Rural Residential Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Conditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	31175						
1971	37291						
1972	43000						
1973	48480						
1974	44807						
1975	39050						
1976	41278						
1977	49582						
1978	57123						
1979	60932						
1980	73200						
1981	73443						
1982	85386						
		Conditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	70129	Sales	Error	Sales	Error	Sales	Error
1984	78993	70075	-11.3	76734	-2.9	73715	-7.2
1985	80093	71355	-10.8	74484	-7.0	76173	-5.1
1986	79157	72590	-8.3	78059	-1.4	78740	-0.5
1987	83672	73809	-11.8	78260	-6.5	81222	-3.0
1988	87814	75014	-14.6	80546	-8.3	83567	-5.1
1989	91063	76190	-16.3	81030	-11	85790	-6.1
1990	96775	77350	-20.1	82713	-14.5	87904	-10.1
Avg.			14.4		9.0		5.8

^a (MWh)

IV.4.4 Rural Commercial Sales Model

Three explanatory (exogenous) input variables are selected to model and forecast the sales to the rural commercial customers. These are year, the number of customer in the service area and the adjusted price of electricity.

Figure 4.21 is a graph of the time series of historical and the forecasted number of rural commercial customers. Table 4.21 shows the historical number of city commercial customers from fiscal year 1970 to 1990 as well as the number of commercial customers forecasted by the City of Bryan from fiscal year 1984 to 1990. The data for the forecasted commercial customers were obtained from the City of Bryan 1984 System Load Forecast [13]. Despite the steady increase in the number of customers during the period 1975 to 1982, the number of commercial customers remained almost constant during the forecasting period of 1986 to 1990.

Figure 4.22 is a graph of the historical and the forecasted adjusted price of electricity for the rural commercial customers in cents per KWh. Table 4.22 shows the historical adjusted price of electricity from fiscal year 1970 to 1990 as well as the forecasted adjusted price of electricity from 1984 to 1990. The drop in the adjusted price of electricity during the period 1985 to 1990 was mainly due to the increase in the Consumer Price Index (CPI), while the actual price of electricity remained constant.

To model and forecast the energy sales in the rural commercial class using NN-1 and NN-2 nonlinear forecasting methods, the aforementioned three time series as well as the sales in MWh were preprocessed. The preprocessing parameters were selected to ensure that the historical values and the forecasted values the exogenous and endogenous variables fall within the linear region of the nonlinear activation functions.

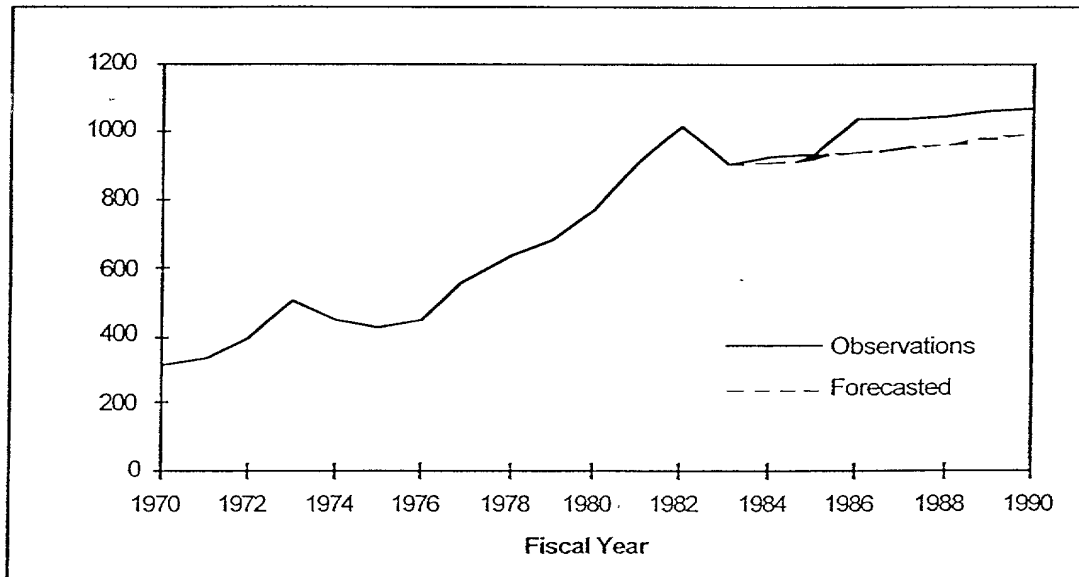


Figure 4.21: Rural Commercial Customers Time Series for the City of Bryan.

Table 4.21: Rural Commercial Customers Time Series for the City of Bryan.

Fiscal Year	Customers	Forecasted Customer ^a	Error (%)
1970	313		
1971	336		
1972	403		
1973	500		
1974	456		
1975	421		
1976	446		
1977	553		
1978	640		
1979	681		
1980	772		
1981	911		
1982	1014		
1983	898		
1984	917	908	-1.0
1985	931	922	-1.0
1986	1033	935	-9.5
1987	1034	949	-8.2
1988	1050	962	-8.4
1989	1051	975	-7.2
1990	1069	988	-7.6

^a Used in conditional forecasting

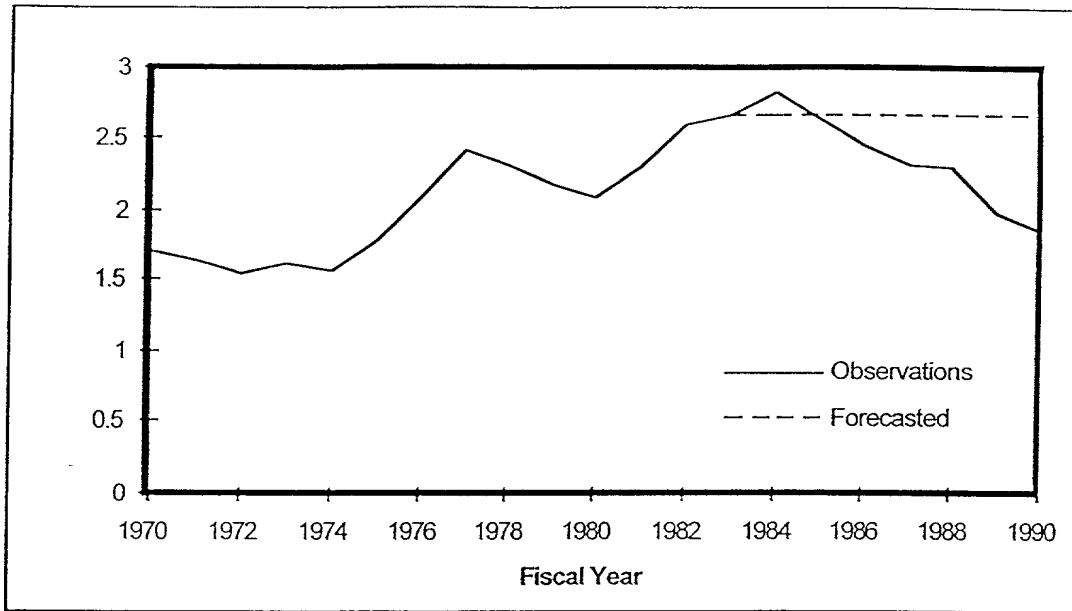


Figure 4.22: Rural Commercial Adjusted Price of Electricity Time Series for the City of Bryan.

Tables 4.23 and 4.24 show the preprocessing parameters and the results of the Monte Carlo filtering process, using NN-1 and NN-2 forecasting methods. The historical data for period 1970 to 1980 was used as the training set and the historical data for period 1981 through 1983 was used as the validation set. The models developed based on 14 years of historical data, were used to forecast the energy sales for a period of 7 years, from 1984 through 1990.

Using the NN-1 forecasting method, the Monte Carlo filtering process selected model structures with one past output with 3 hidden nodes, 2 past outputs with 2 hidden nodes and 2 past outputs with 4 hidden nodes. Initial seed numbers of top performing models of each model structure are shown in Table 4.23. Using the NN-2 forecasting method, the Monte Carlo filtering process selected model structures with one

Table 4.22: Rural Commercial Adjusted Price of Electricity Time Series for the City of Bryan.

Fiscal Year	Adjusted Price (c/kwh)	Forecasted Adjusted Price (c/kwh) ^a	Error (%)
1970	1.72		
1971	1.64		
1972	1.55		
1973	1.62		
1974	1.56		
1975	1.79		
1976	2.08		
1977	2.41		
1978	2.31		
1979	2.17		
1980	2.08		
1981	2.29		
1982	2.6		
1983	2.67		
1984	2.82	2.67	-5.3
1985	2.61	2.67	2.3
1986	2.45	2.67	9.0
1987	2.3	2.67	16.1
1988	2.29	2.67	16.6
1989	1.99	2.67	34.2
1990	1.85	2.67	44.3

^a Used in conditional forecasting

Table 4.23: City of Bryan Rural Commercial NN-1 Forecasting Model Specifications.

Data Transformations			
Time	$\frac{Year - 1969}{21}$		
Customers	$\frac{Customers - 1200}{1200}$		
Adjusted Price	$\frac{Adjusted Price - 1.5}{2.0}$		
Sales	$\frac{Sales - 60000}{60000}$		
Training Parameters			
Learning Rate	0.01		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Training Error	5%		
Max. Number of Iterations Allowed	10000		
Forecasting Model Structures			
Number of Past Outputs	1	2	2
Number of Hidden Nodes	3	2	4
Top Performing NN-1 Models	Seed Numbers		
1	149170	176978	302690
2	83602	248434	51574
3	164342	126818	303058
4	46770	301158	430774
5	231698	71942	401282
6	8018	412166	4998
7	228870	91362	455058
8	89442	263522	486722
9	228070	190802	385734
10	519266	387874	57394

Table 4.24: City of Bryan Rural Commercial NN-2 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	21		
Customers	<u>Customers</u>		
	1200		
Adjusted Price	<u>Adjusted Price - 1.5</u>		
	2.0		
Sales	<u>Sales</u>		
	60000		
Training Parameters			
Learning Rate	0.1		
Momentum Coefficient	0.3		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Max. Number of Iterations Allowed	10000		
Forecasting Model Structures			
Number of Past Outputs	1	1	2
Number of Hidden Nodes	2	1	1
Top Performing NN-2 Models	Seed Numbers		
1	83319	209631	485271
2	82485	695	38647
3	167575	152671	509239
4	217567	301775	211511
5	140405	411135	492983
6	34127	136613	482775
7	127949	97767	420375
8	490781	174949	119895
9	115917	295541	256855
10	301759	306549	129207

node. Initial seed numbers of top performing models of each model structure are also shown in Table 4.24.

From Tables 4.23 and 4.24, it is clear that different stopping criteria resulted in the selection of different model structures. Using the NN-2 forecasting method, the average error and variance of the top performing models for each of the best 3 model structures are (0.25, 0.03), (0.27, 0.04) and (0.32, 0.04), respectively. Using the NN-1 forecasting technique, the best generalization capability was obtained when the training process was terminated at 5% error levels. However, ~~using~~ the NN-2 forecasting technique, the crossing between the training set and the validation set error profiles occurred at a much higher error level, around 25%.

In 1984 the City of Bryan developed a forecast model for energy sales to its rural commercial customers [13]. This model is a linear regression model and is given in equation (4.4). The primary contributors to energy consumption in the city commercial class are the number of customers and the price of electricity. In addition to these explanatory variables, a “crisis” variable is used to explain major discontinuities in the historical data which can not be explained by the main variables. This variable has a value of 1 from 1970 through 1982, and 0 thereafter.

$$\begin{aligned} \text{Sales} = & 8605.63913 + 39.6612806 (\text{No. of Customers}) \\ & - 3059.7263 (\text{Adjusted Price}) - 5413.9274 (\text{Crisis}) \end{aligned} \quad (4.4)$$

Figures 4.23 and 4.24 show the unconditional forecast and forecast errors of the energy sales in the rural commercial class. The historical energy sales and numerical values of unconditional forecasts and forecast errors are also shown in Table 4.25. In developing the unconditional Neural Network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System

Load forecast [13]. The average error of City of Bryan utility forecast, NN-1 and NN-2 are 21.8%, 14.6% and 14.5%, respectively. The City of Bryan forecast underestimated

2025 RELEASE UNDER E.O. 14176

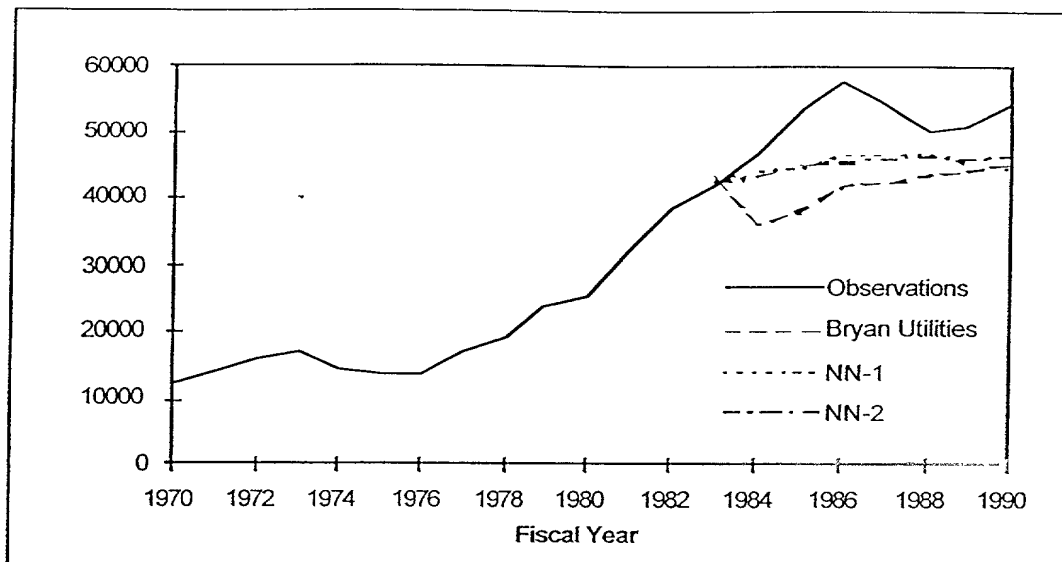


Figure 4.23: Unconditional Forecast of the Rural Commercial Energy Sales for the City of Bryan.

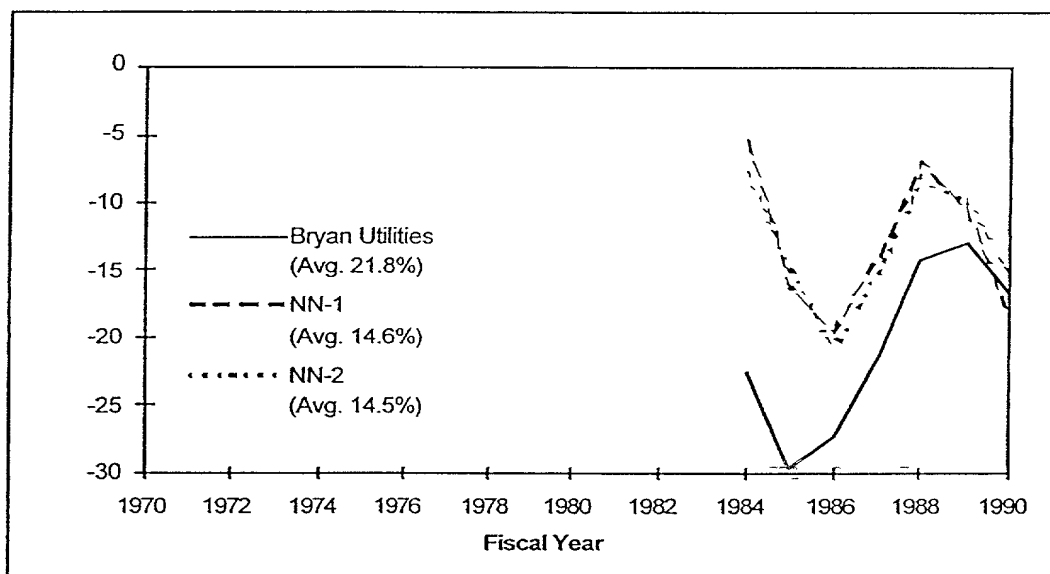


Figure 4.24: Unconditional Forecast Errors of the Rural Commercial Energy Sales for the City of Bryan.

Table 4.25: Unconditional Energy Sales Forecasts and Forecasting Errors of the Rural Commercial Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Unconditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	12615						
1971	13971						
1972	15880						
1973	17014						
1974	14351						
1975	13678						
1976	13833						
1977	16970						
1978	19471						
1979	23715						
1980	25570						
1981	32579						
1982	38280						
		Unconditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	42520	Sales	Error	Sales	Error	Sales	Error
1984	46954	36347	-22.6	44271	-5.7	43218	-8
1985	53437	37544	-29.7	44647	-16.4	45313	-15.2
1986	57847	42079	-27.3	46399	-19.8	45576	-21.2
1987	54091	42578	-21.3	46491	-14.1	45832	-15.3
1988	50381	43243	-14.2	46767	-7.2	46259	-8.2
1989	50823	44201	-13	45441	-10.6	45814	-9.9
1990	54420	45343	-16.7	44461	-18.3	46173	-15.2

^a (MWh)

Avg.			21.8		14.6		14.5
------	--	--	------	--	------	--	------

the sales in 1984 by 22.6% and NN-1 and NN-2 underestimated the sales in 1984 by 5.7% and 8% respectively. The maximum annual unconditional forecast error for the City of Bryan 1984 System Load Forecast is 29.7% in 1985. The maximum annual unconditional forecast error for NN-1 is 19.8% in 1986 and for NN-2 is 21.2% in 1986.

Figures 4.25 and 4.26 show the conditional forecast and forecast errors of the energy sales in the rural commercial class. The historical energy sales and numerical values of conditional forecasts and forecast errors are also shown in Table 4.26. In developing the conditional Neural Network forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. The projections of exogenous variables used in the City of Bryan 1984 System Load forecast [13] were used to generate conditional forecasts. The average forecast error for the City of Bryan 1984 System Load forecast was 28.2% compared to 15.1% and 13.9% for NN-1 and NN-2 respectively. The City of Bryan 1984 System Load Forecast underestimated the 1984 energy sales by 22.4%. The NN-1 underestimated the 1984 energy sales by 7.5%, while the NN-2 overestimated the 1990 sales by 9%. The maximum annual conditional forecast error for the City of Bryan 1984 System Load Forecast is 35.1% in 1986. The maximum annual conditional forecast error for NN-1 is 22.2% in 1986 and for NN-2 is 21.8% in 1986.

The rural commercial class has experienced a structural change during 1980's. While the number of commercial customers remained almost constant, the energy sales in this class continued to increase.

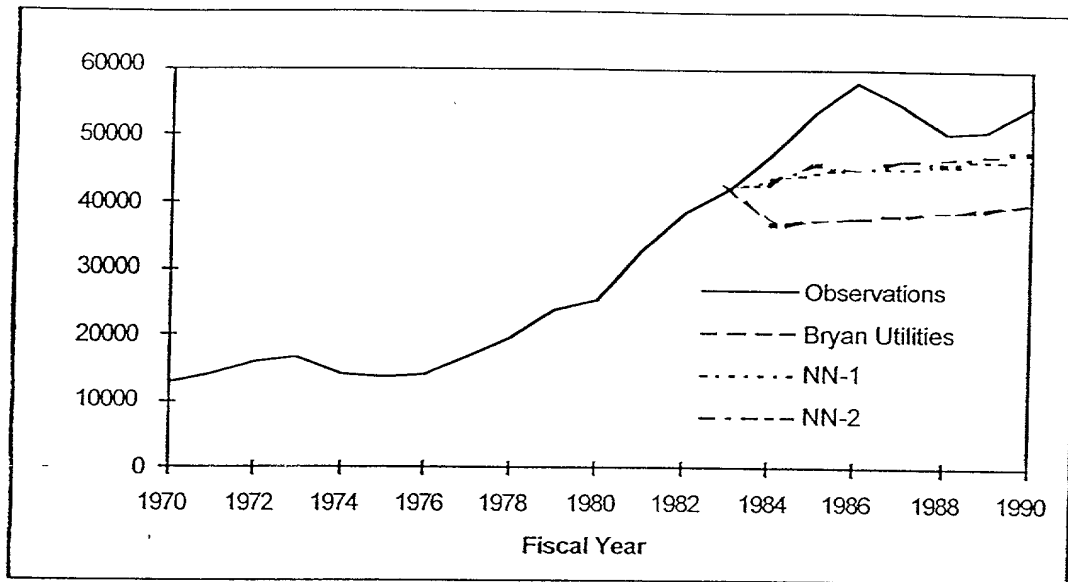


Figure 4.25: Conditional Forecast of the Rural Commercial Energy Sales for the City of Bryan.

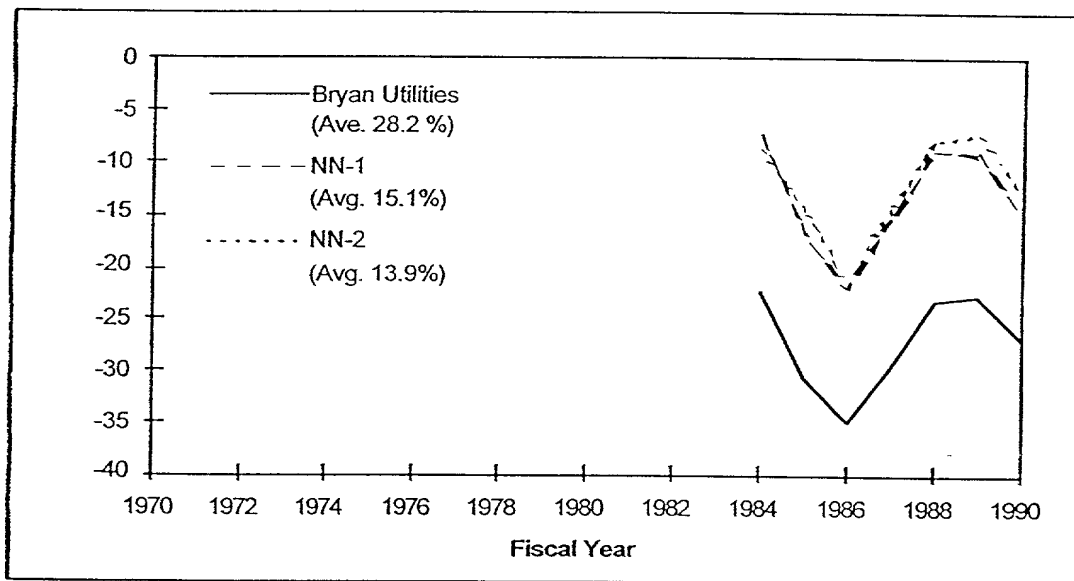


Figure 4.26: Conditional Forecast Errors of the Rural Commercial Energy Sales for the City of Bryan.

Table 4.26: Conditional Energy Sales Forecasts and Forecasting Errors of the Rural Commercial Class for the City of Bryan.

Fiscal Year	Energy Sales ^a	Conditional Forecasted Energy Sales ^a and Forecasting Errors (%)					
1970	12615						
1971	13971						
1972	15880						
1973	17014						
1974	14351						
1975	13678						
1976	13833						
1977	16970						
1978	19471						
1979	23715						
1980	25570						
1981	32579						
1982	38280						
		Conditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	42520	Sales	Error	Sales	Error	Sales	Error
1984	46954	36456	-22.4	43418	-7.5	42717	-9
1985	53437	37006	-30.7	44337	-17	45597	-14.7
1986	57847	37538	-35.1	44984	-22.2	45244	-21.8
1987	54091	38069	-29.6	45528	-15.8	46193	-14.6
1988	50381	38594	-23.4	45911	-8.9	46415	-7.9
1989	50823	39113	-23	46170	-9.2	47134	-7.3
1990	54420	39626	-27.2	46330	-14.9	47510	-12.7

^a (MWh)

Avg.			28.2		15.1		13.9
------	--	--	------	--	------	--	------

IV.4.5 City of Bryan Peak Load and Total System Demand Model

The system peak load demand for the City of Bryan experienced a steady growth during the period 1970 to 1983. However, the system peak load demand has not increased significantly since the year 1983. Among other factors, this could be attributed to improvements in the load factor and/or the demographic changes in the service area.

In order to model and forecast the peak load demand in the city of Bryan, a nonlinear NN econometric model was developed. After examining several explanatory input variables, three inputs were selected namely the time, the energy sales to the city residential class of the city and CDDs. The historical and the forecasted energy sales to the city residential class are shown in Figures 4.5 and 4.7 and the CDDs for the City of Bryan are shown in Figure 4.3.

In 1984 the City of Bryan forecasted system peak load demand by first forecasting total energy sales and total system demand, and then applying a load factor, assumed constant, to estimate the peak load demand. It is not possible to generate an unconditional forecast of the peak load demand, using this method

To model the peak load demand for the City of Bryan using the proposed nonlinear forecasting technique, the aforementioned three time series, as well as the peak load demand in MW were preprocessed using the transformations shown in Table 4.27 and Table 4.28. The preprocessing parameters were selected to ensure that the historical values and the forecasted values of the exogenous and endogenous variables fall within the linear region of the nonlinear activation functions.

Tables 4.27 and 4.28 show the preprocessing transformations and the results of the Monte Carlo filtering process using NN-1 and NN-2 forecasting methods. The historical data for period 1970 to 1981 was used as the training set and the historical data for period 1982 through 1984 was used as the validation set. The models developed

Table 4.27: City of Bryan Peak Load Demand NN-1 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	21		
Residential Sales	<u>Re sidential Sales</u>		
	250000		
CDD	<u>CDD - 2000</u>		
	1200.		
Peak Load Demand	<u>Peak Load Demand</u>		
	190		
Training Parameters			
Learning Rate	0.003		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Training Error	5%		
Max. Number of Iterations Allowed	7000		
Forecasting Model Structures			
Number of Past Outputs	1	1	2
Number of Hidden Nodes	3	4	4
Top Performing NN-1 Models	Seed Numbers		
1	52850	3190	516178
2	262694	468598	336882
3	166386	219634	205270
4	442882	215478	269206
5	522262	142726	303058
6	503766	322598	105458
7	337922	476818	224994
8	513014	360882	64290
9	325158	360790	157814
10	384850	332230	357830

Table 4.28: City of Bryan Peak Load Demand NN-2 Forecasting Model Specifications.

Data Transformations			
Time	<u>Year - 1969</u>		
	21		
Residential Sales	<u>Residential Sales</u>		
	250000		
CDD	<u>CDD - 2000</u>		
	1200		
Peak Load Demand	<u>Peak Load Demand</u>		
	190		
Training Parameters			
Learning Rate	0.1		
Momentum Coefficient	0.3		
Estimation Period	1970 - 1983		
Training Period	1970 - 1980		
Validation Period	1981 - 1983		
Forecasting Period	1984 - 1990		
Max. Number of Iterations Allowed	5000		
Forecasting Model Structures			
Number of Past Outputs	1	2	1
Number of Hidden Nodes	3	3	2
Top Performing NN-2 Models	Seed Numbers		
1	129826	187698	304514
2	255398	305026	418706
3	489670	252354	325106
4	443926	460550	202146
5	36210	142294	347778
6	454226	26258	230562
7	439410	307910	38466
8	315666	385414	151202
9	3638	335350	408258
10	416738	327122	30518

based on 14 years of historical data, were used to forecast the peak load for a period of 7 years, from 1984 through 1990.

Using NN-1 forecasting method, the Monte Carlo filtering process selected model structures with one past output with 3 hidden nodes, 1 past output with 4 hidden nodes and 2 past outputs with 4 hidden nodes. Initial seed numbers of top performing models of each model structure are shown in Table 4.27. Using NN-2 forecasting method, the filtering process selected model structures with 1 past output with 3 hidden nodes, 2 past outputs with 3 hidden nodes and 1 past output with 2 hidden nodes. Initial seed numbers of top performing models of each model structure are also shown in Table 4.28.

From Tables 4.27 and 4.28 it is clear that different stopping criteria resulted in the selection of different model structures. Using NN-2 forecasting method, the average error and variance of the top performing models for each of the best 3 model structures are (0.057, 0.008), (0.072, 0.022) and (0.089, 0.029), respectively. Using NN-1 forecasting method, the best generalization capability was obtained when the training was terminated at 5% error level. However, using NN-2 forecasting method the crossing point between the training and the validation set has occurred at about 5.7% error level.

Figures 4.27 and 4.28 show the unconditional forecast and forecast errors of the peak load demand in the City of Bryan, using NN-1 and N-2 forecasting methods. The City of Bryan 1984 System Load Forecast [13] has used a constant load factor method to generate the peak load forecast. Using this technique, it is not possible to develop an unconditional forecast of the peak load demand. The historical peak load demand time series and numerical values of unconditional forecasts and forecast errors are also shown in Table 4.29. In developing the unconditional neural networks forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of

Bryan 1984 System Load forecast [13]. The average unconditional forecast error for NN-1 and NN-2 are 4.1% and 5.3% respectively.

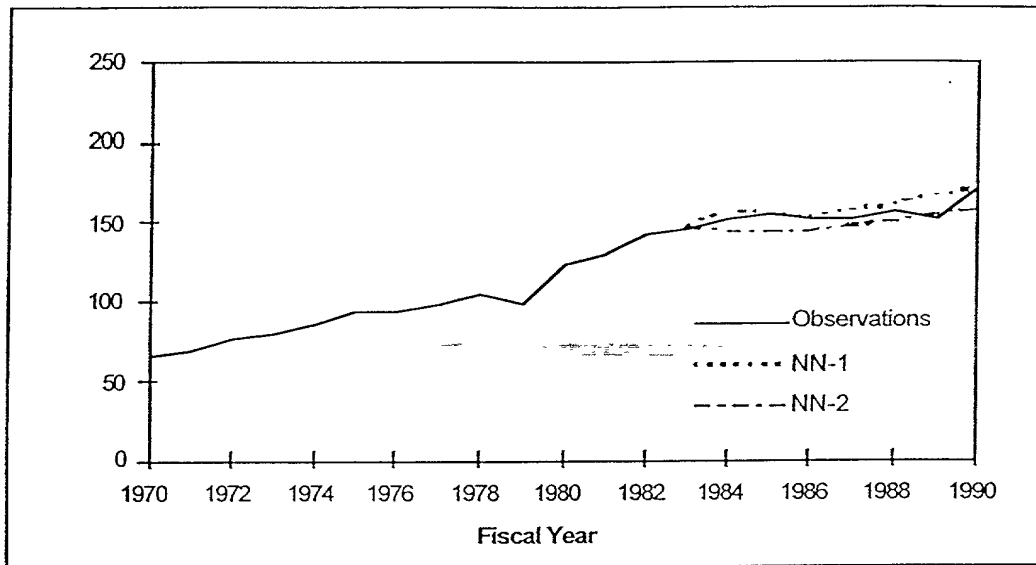


Figure 4.27: Unconditional Forecast of the Peak Load Demand for the City of Bryan.

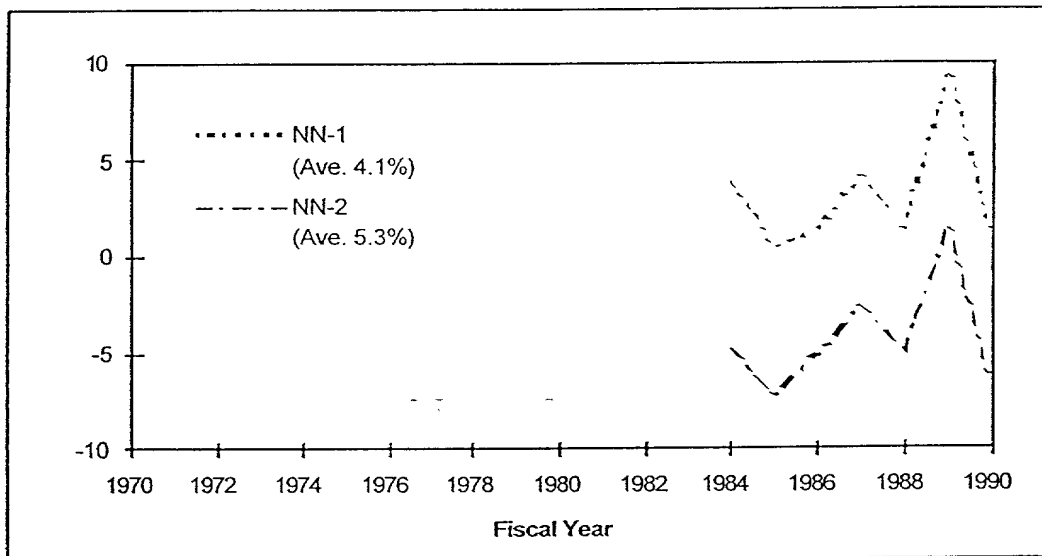


Figure 4.28: Unconditional Forecast Error of the Peak Load Demand for the City of Bryan.

Table 4.29: Unconditional Peak Load Demand Forecasts and Forecast Error for the City of Bryan.

Fiscal Year	Peak Load Demand	Unconditional Forecasted Peak Demand and Forecasting Error (%)			
1970	66.1				
1971	70.4				
1972	77.5				
1973	81.4				
1974	86.9				
1975	92.9				
1976	92.9				
1977	99.0				
1978	104.4				
1979	99.2				
1980	125.2				
1981	128.8				
1982	143.0				
1983	146.9	Unconditional Forecast			
		NN-1		NN-2	
		Demand	Error	Demand	Error
1984	152.0	157.4	3.6	144.3	-5.1
1985	155.0	155.9	0.5	143.7	-7.3
1986	152.0	154.2	1.4	143.9	-5.3
1987	152.0	158.0	3.9	147.9	-2.7
1988	158.0	160.2	1.4	150.0	-5.1
1989	153.0	167.3	9.3	155.2	1.4
1990	170.0	171.6	1.0	158.2	-6.9
Avg.			4.1		5.3

The NN-1 overestimated the peak load demand throughout the forecasting period, while NN-2 underestimated the peak load demand in most of the years. The average unconditional forecasting error for NN-1 and NN-2 forecasting methods are 4.1% and 5.3%, respectively. The NN-1 forecasting method overestimated the peak load demand in 1984 by 3.6%, while NN-2 underestimated the peak load demand in 1984 by 5.1%. The maximum annual unconditional forecast error for NN-1 is 9.3% in 1989 and for NN-2 is 6.9% in 1990.

Figures 4.29 and 4.30 show the conditional forecast and forecast errors of peak load demand for the City of Bryan. The historical peak load and numerical values of conditional forecasts and forecast errors are also shown in Table 4.30. In developing the conditional neural networks forecasts, the historical observations for years 1970 to 1983 has been used for consistency with the City of Bryan 1984 System Load forecast [13]. To generate conditional forecasts, the conditional forecast of the energy sales in the city residential class of City of Bryan was used. These forecasts are shown in Figure 4.8 and Table 4.7. The City of Bryan 1984 System Load forecast [13] has forecasted the peak load demand by forecasting the total system demand initially. Using a constant load factor equal to the load factor in the year 1983, the peak load demand was forecasted based on the total system energy demand.

While all forecasts overestimated the peak load demand, the average error for the City of Bryan 1984 System Load forecast was 18.1% compared to 11.8% and 6.4% for NN-1 and NN-2 respectively. In the year 1984, the City of Bryan 1984 overestimated the 1984 peak load demand by 3% while NN-1 and NN-2 underestimated the peak load demand in 1984 by 1.9% and 1.4%, respectively. The maximum annual conditional forecast error for the City of Bryan 1984 System Load Forecast is 29.0% in 1988. The

maximum annual conditional forecast error for NN-1 is 20.7% in 1989 and for NN-2 is 12.4% in 1989.

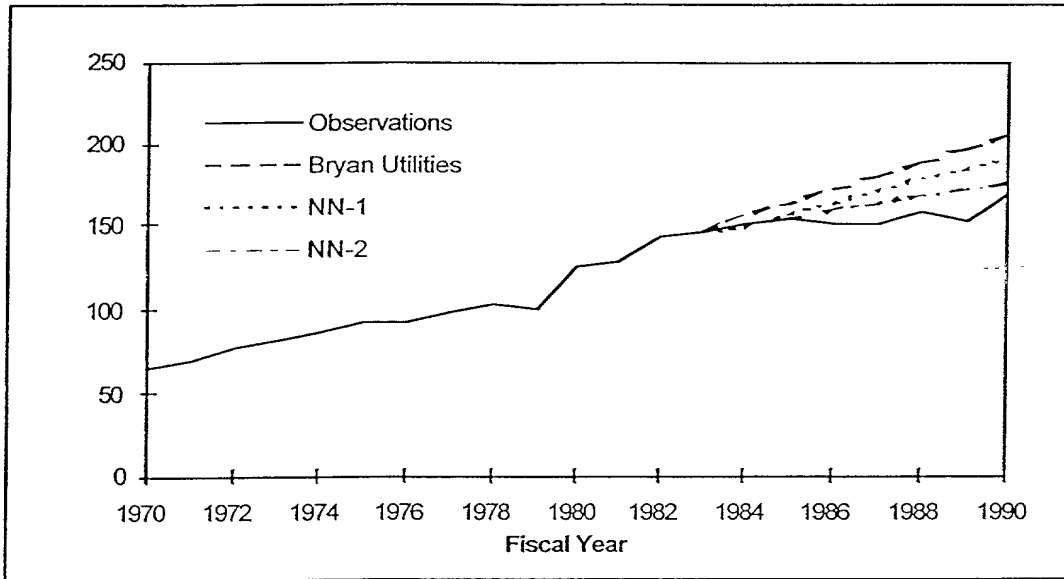


Figure 4.29: Conditional Forecast of the Peak Load Demand for the City of Bryan.

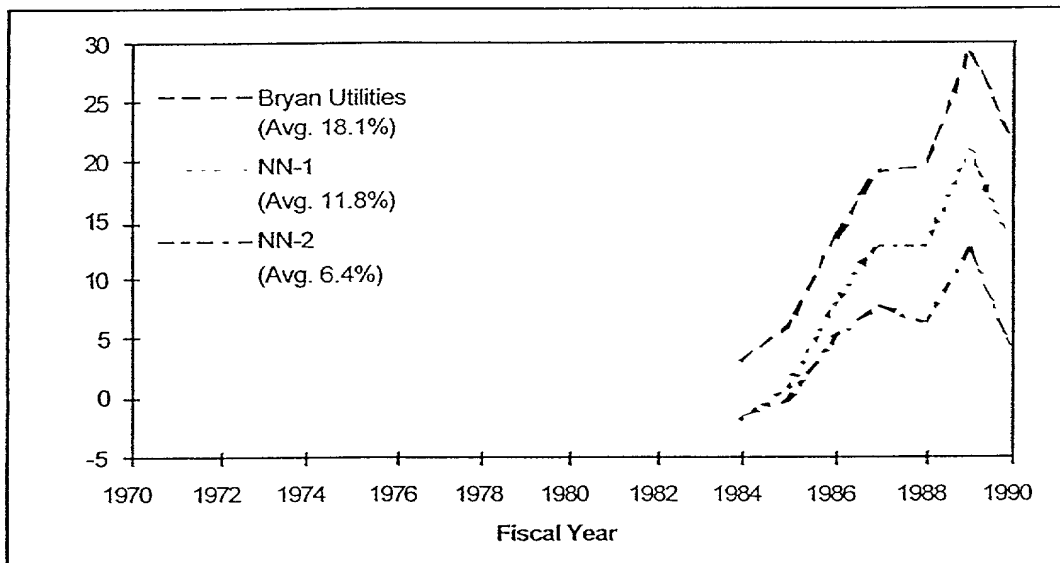


Figure 4.30: Conditional Forecast Error of the Peak Load Demand for the City of Bryan.

	231 ^a	232 ^a	233 ^a	234 ^a	235 ^a	236 ^a	237 ^a	238 ^a	239 ^a	240 ^a	241 ^a	242 ^a	243 ^a	244 ^a	245 ^a	246 ^a	247 ^a	248 ^a	249 ^a	250 ^a	251 ^a	252 ^a	253 ^a	254 ^a	255 ^a	256 ^a	257 ^a	258 ^a	259 ^a	260 ^a	261 ^a	262 ^a	263 ^a	264 ^a	265 ^a	266 ^a	267 ^a	268 ^a	269 ^a	270 ^a	271 ^a	272 ^a	273 ^a	274 ^a	275 ^a	276 ^a	277 ^a	278 ^a	279 ^a	280 ^a	281 ^a	282 ^a	283 ^a	284 ^a	285 ^a	286 ^a	287 ^a	288 ^a	289 ^a	290 ^a	291 ^a	292 ^a	293 ^a	294 ^a	295 ^a	296 ^a	297 ^a	298 ^a	299 ^a	300 ^a	301 ^a	302 ^a	303 ^a	304 ^a	305 ^a	306 ^a	307 ^a	308 ^a	309 ^a	310 ^a	311 ^a	312 ^a	313 ^a	314 ^a	315 ^a	316 ^a	317 ^a	318 ^a	319 ^a	320 ^a	321 ^a	322 ^a	323 ^a	324 ^a	325 ^a	326 ^a	327 ^a	328 ^a	329 ^a	330 ^a	331 ^a	332 ^a	333 ^a	334 ^a	335 ^a	336 ^a	337 ^a	338 ^a	339 ^a	340 ^a	341 ^a	342 ^a	343 ^a	344 ^a	345 ^a	346 ^a	347 ^a	348 ^a	349 ^a	350 ^a	351 ^a	352 ^a	353 ^a	354 ^a	355 ^a	356 ^a	357 ^a	358 ^a	359 ^a	360 ^a	361 ^a	362 ^a	363 ^a	364 ^a	365 ^a	366 ^a	367 ^a	368 ^a	369 ^a	370 ^a	371 ^a	372 ^a	373 ^a	374 ^a	375 ^a	376 ^a	377 ^a	378 ^a	379 ^a	380 ^a	381 ^a	382 ^a	383 ^a	384 ^a	385 ^a	386 ^a	387 ^a	388 ^a	389 ^a	390 ^a	391 ^a	392 ^a	393 ^a	394 ^a	395 ^a	396 ^a	397 ^a	398 ^a	399 ^a	400 ^a	401 ^a	402 ^a	403 ^a	404 ^a	405 ^a	406 ^a	407 ^a	408 ^a	409 ^a	410 ^a	411 ^a	412 ^a	413 ^a	414 ^a	415 ^a	416 ^a	417 ^a	418 ^a	419 ^a	420 ^a	421 ^a	422 ^a	423 ^a	424 ^a	425 ^a	426 ^a	427 ^a	428 ^a	429 ^a	430 ^a	431 ^a	432 ^a	433 ^a	434 ^a	435 ^a	436 ^a	437 ^a	438 ^a	439 ^a	440 ^a	441 ^a	442 ^a	443 ^a	444 ^a	445 ^a	446 ^a	447 ^a	448 ^a	449 ^a	450 ^a	451 ^a	452 ^a	453 ^a	454 ^a	455 ^a	456 ^a	457 ^a	458 ^a	459 ^a	460 ^a	461 ^a	462 ^a	463 ^a	464 ^a	465 ^a	466 ^a	467 ^a	468 ^a	469 ^a	470 ^a	471 ^a	472 ^a	473 ^a	474 ^a	475 ^a	476 ^a	477 ^a	478 ^a	479 ^a	480 ^a	481 ^a	482 ^a	483 ^a	484 ^a	485 ^a	486 ^a	487 ^a	488 ^a	489 ^a	490 ^a	491 ^a	492 ^a	493 ^a	494 ^a	495 ^a	496 ^a	497 ^a	498 ^a	499 ^a	500 ^a	501 ^a	502 ^a	503 ^a	504 ^a	505 ^a	506 ^a	507 ^a	508 ^a	509 ^a	510 ^a	511 ^a	512 ^a	513 ^a	514 ^a	515 ^a	516 ^a	517 ^a	518 ^a	519 ^a	520 ^a	521 ^a	522
--	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	------------------	-----

Table 4.30: Conditional Peak Load Demand Forecasts and Forecast Error for the City of Bryan.

Fiscal Year	Peak Load Demand ^a	Conditional Forecasted Peak Demand and Forecasting Error (%)					
1970	66.1						
1971	70.4						
1972	77.5						
1973	81.4						
1974	86.9						
1975	92.9						
1976	92.9						
1977	99.0						
1978	104.4						
1979	99.2						
1980	125.2						
1981	128.8						
1982	143.0	Conditional Forecast					
1983	146.9	Bryan Utilities		NN-1		NN-2	
		Demand	Error	Demand	Error	Demand	Error
1984	152.0	156.6	3.0	149.1	-1.9	149.9	-1.4
1985	155.0	164.6	6.2	156.5	1.0	154.7	-0.2
1986	152.0	172.2	13.6	164.2	8.0	159.4	4.9
1987	152.0	180.8	19.0	171.4	12.8	163.9	7.8
1988	158.0	189.0	19.7	178.3	12.8	168.0	6.3
1989	153.0	197.3	29.0	184.7	20.7	171.9	12.4
1990	170.0	205.6	21.0	190.8	12.2	175.6	3.3
Avg.			18.1		11.8		6.4

^a (MWh)

As expected, performing a conditional forecast results in an increase of the annual forecasting errors for both the NN models and the City of Bryan 1984 System Load Forecasting model, since the error in the forecast of the exogenous variables will also contribute in the total forecasting error. However, NN-1 and NN-2 outperformed the City of Bryan forecast of the peak load demand in the forecasting period.

Figures 4.31 and 4.32 show the conditional forecast of total system energy demand and forecast error for the City of Bryan in MWh. The numerical values of the historical total energy demand time series as well as the conditional forecasts and forecast errors are also shown in Table 4.31. The average historical values of the line losses for the city system and the rural system were 5.6% and 9.8% of their total energy demand respectively. Adding line losses to the total energy sales results in the total energy demand for the city and rural system. This is further adjusted by accounting an additional 5% of the total energy demand for school and interdepartmental classes. The City of Bryan Utilities Staff, NN-1 and NN-2 forecasted the total energy demand in the City of Bryan with an average error of 4.5%, 2.9% and 2%, respectively. The City of Bryan 1984 load forecast [13] underestimated the total energy demand in 1984 by 5.3%, while NN-1 and NN-2 underestimated the energy demand in 1984 by 5.2% and 3.2% respectively. The maximum annual forecast error of the total system demand for City of Bryan, NN-1 and NN-2 are 5.8% in 1989, 5.6% in 1985 and 3.2% in 1984, respectively.

NN-1 and NN-2 outperformed the City of Bryan in forecasting total system demand during the forecasting period.

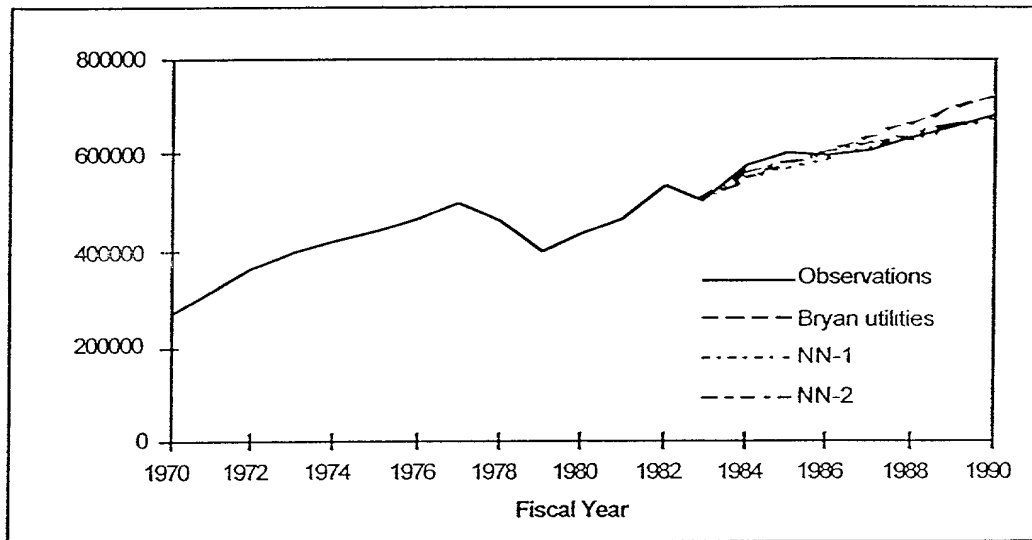


Figure 4.31: Conditional Forecast of System Total Demand for the City of Bryan.

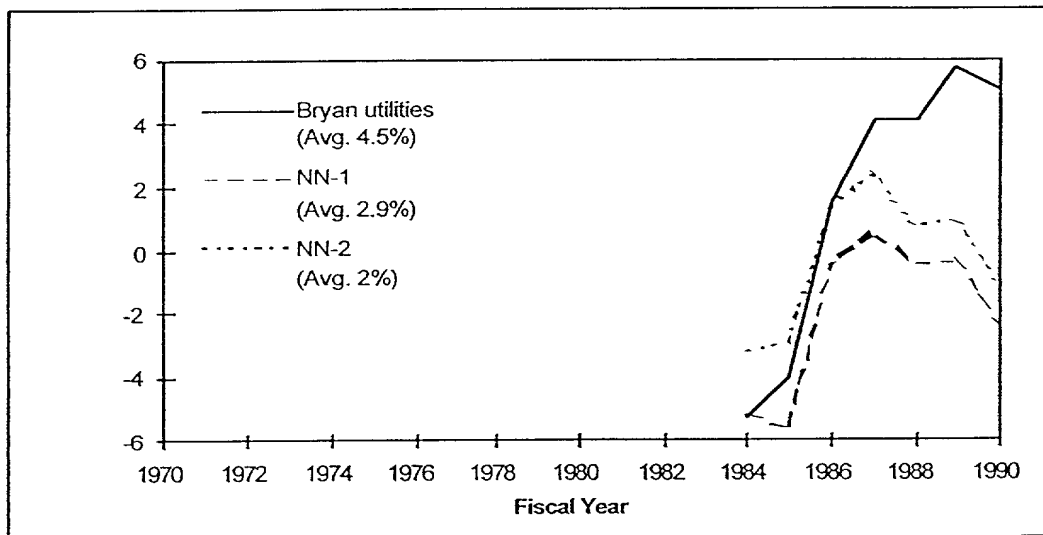


Figure 4.32: Conditional Forecast Error of System Total Demand for the City of Bryan.

Table 4.31: Conditional Forecasts of Total System Demand for the City of Bryan.

Fiscal Year	Energy Demand ^a	Conditional Forecasted Energy Demand ^a and Forecasting Errors (%)					
1970	270716						
1971	312131						
1972	362244						
1973	398988						
1974	421316						
1975	445985						
1976	470523						
1977	495554						
1978	465518						
1979	396478						
1980	438547						
1981	469946						
1982	540443						
		Conditional Forecast					
		Bryan Utilities		NN-1		NN-2	
1983	502248	Demand	Error	Demand	Error	Demand	Error
1984	580246	549360	-5.3	550070	-5.2	561843	-3.2
1985	602240	577543	-4.1	568496	-5.6	584514	-2.9
1986	596105	605878	1.6	593217	-0.5	604002	1.3
1987	609349	634426	4.1	612863	0.6	623885	2.4
1988	636963	663189	4.1	634101	-0.4	642354	0.8
1989	654209	692158	5.8	652146	-0.3	660804	1.0
1990	686240	721356	5.1	670659	-2.3	678306	-1.2
Avg.			4.5		2.9		2.0

^a (MWh)

IV.5 TMPA Forecasting Summary

The proceeding sections have described in detail the results of load forecasts using alternative models for the City of Bryan. Similar studies were also carried out for the cities of Denton, Garland and Greenville. These studies were carried out using data and past forecasts as available from cities.

Cities of Denton provided a reference forecast performed sometime in the past. The NN-1 and NN-2 forecasting methods used only those historical observations and forecasts that was available to the forecaster at the time the reference forecast. However, City of Denton did not provide any information about the underlying forecasting models or selected exogenous variables. Therefore, no unconditional forecasts were performed for the City of Denton.

City of Garland did not provide any forecasts of sales in different customer classes nor any information about the underlying forecasting models or exogenous variables. We set aside several years of historical observations as the forecasting period. Using the provided time series of the possible exogenous variables, four energy sales models were developed using NN-1 and NN-2 forecasting methods. Using the forecasts of sales in different classes of customers a forecast of total system demand in the City of Garland was developed. Finally, peak load demand in the City of Garland was forecasted using NN-1 and NN-2 forecasting methods. The comparison in this city is between NN-1 and NN-2 forecasting methods.

For the particular case of the City of Greenville, individual customer class energy sales models, and utility supplied forecasts for the individual customer classes were not available. Furthermore, the available historical time-series were extremely short.

Therefore, for these models no historical observations or other forecasts were available for comparison purposes during the forecasting period.

A summary of the load forecasting results for the four TMPA cities is given in Tables 4.32 through 4.35. These tables show the average and the maximum annual forecast errors for the four utilities for the two neural network models and for the utility conducted forecasts if available. Tables 4.32 and 4.33 refer to unconditional energy sales and peak load demand forecasts, respectively, whereas Tables 4.34 and 4.35 refer to the conditional energy sales and peak load demand forecasts, respectively.

The following remarks can be made about the unconditional forecasts based on the information presented in Tables 4.32 and 4.33:

1. Energy sales and peak load demand forecasts produced using the new neural network models consistently exhibit improved accuracy over the utility conducted forecasts. This is true both in terms of average and maximum forecast error during the study period.
2. The improvement in the average forecast error for energy sales ranges from 10% up to 49% for the NN-1 neural network models and compared to the utility conducted forecast. The improvement in the average forecast error for energy sales ranges from 30% up to 70% for the NN-2 neural network model and compared to the utility conducted forecast. In about 70% of the developed models, the NN-2 model has a lower average forecast error for energy sales compared to the NN-1 model.
3. With the exception of the commercial customer classes of the City of Bryan, the average NN-1 model forecast errors for energy sales are less than 10% and the maximum forecast errors for energy sales are less than 14%. With the

exception of the commercial customer classes of the City of Bryan, the average NN-2 model forecast errors for energy sales are less than 7.7% and the maximum forecast errors for energy sales are less than 11.3%.

2025-2026 Energy Sales Forecast

Table 4.32 Summary of Unconditional Energy Sales Forecasts by Customer Class.

City	Customer Class	Forecast Period	Average (MSE) Annual Forecast Error (%)			Maximum Annual Forecast Error (%)		
			Utilit	NN-	NN-	Utilit	NN-	NN-
			y- Staff	1 -	2 -	y- Staff	1 -	2 -
Bryan	Residential ^a	1984-1990	4.4	3.6	3.6	-9.1	6.3	-7.7
	Commercial	1984-1990	15.7	14.1	10.5	-20.1	-19.5	14.5
	Residential ^b	1984-1990	12.8	6.5	3.7	-17.1	-10.8	-6.2
	Commercial ^b	1984-1990	21.8	14.6	14.5	-29.7	-19.8	-21.2
Denton	Residential	1986-1992	— ^c	9.1	6.5	— ^c	-13.9	9.6
	Commercial	1986-1992	— ^c	— ^d	— ^d	— ^c	— ^d	— ^d
Garland	Residential ^e	1988-1992	— ^c	3.6	3.8	— ^c	-5.1	5.6
	Residential ^f	1988-1992	— ^c	3.3	6.3	— ^c	-5.0	-9.1
	Gen. Service	1988-1992	— ^c	8.1	6.2	— ^c	-12.4	-8.0
	Public Sector	1988-1992	— ^c	4.8	7.7	— ^c	-7.3	-11.3
Greenville	Residential	1994-1998	— ^g	— ^g	— ^g	— ^g	— ^g	— ^g
	Gen. service	1994-1998	— ^g	— ^g	— ^g	— ^g	— ^g	— ^g
	Total	1989-1993	— ^c	— ^d	— ^d	— ^c	— ^d	— ^d

^a City^b Rural^c Forecasting model needed to generate unconditional forecasts not supplied by utility staff.^d Insufficient data; NN time-series models do not allow generation of unconditional forecasts.^e Customers using gas.^f Customers using electricity.^g Not available; forecast period not appropriate for error calculations.

Table 4.33 Summary of Unconditional Total System Peak Load Demand Forecasts.

City	Forecast Period	Average (MSE) Annual Forecast Error (%)			Maximum Annual Forecast Error (%)		
		Utilit y Staff	NN- 1	NN- 2	Utilit y Staff	NN- 1	NN- 2
Bryan	1984-1990	— ^a	4.1	5.3	— ^a	9.3	-7.3
Denton	1986-1992	— ^b	4.4	4.9	— ^b	8.5	6.7
Garland	1988-1992	— ^b	2.2	4.6	— ^b	3.0	7.1
Greenville	1989-1993	— ^b	5.0	7.0	— ^b	9.4	12.3

4. The average NN-1 model forecast errors for peak load demand are less than 5% and the maximum forecast errors for peak load demand are less than 10%. The average NN-2 model forecast errors for peak load demand are less than 7.0% and the maximum forecast errors for peak load demand are less than 12.3%.
5. Considering the 9 energy sales neural network models presented in Table 4.32, using NN-1 and NN-2 methods, there was no clear winner between NN-2 and NN-1 forecasting models.

The following remarks can also be made about the conditional forecasts based on the information presented in Tables 4.34 and 4.35:

^a Constant load factor model cannot generate unconditional forecasts.

Table 4.34 Summary of Conditional Energy Sales Forecasts by Customer Class.

City	Customer Class	Forecast Period	Average (MSE) Annual Forecast Error (%)			Maximum Annual Forecast Error (%)		
			Utilit	NN-	NN-	Utilit	NN-	NN-
			y Staff	1	2	-y Staff	1	2
Bryan	Residential ^a	1984-1990	15.9	8.6	5.9	22.4	10.8	-9.4
	Commercial ^a	1984-1990	7.9	10.1	7.1	-11.3	-13.0	9.8
	Residential ^b	1984-1990	14.4	9.0	5.8	-20.1	-14.5	-10.1
	Commercial ^b	1984-1990	28.2	15.1	13.9	-35.1	-22.2	-21.8
Denton	Residential	1986-1992	9.6	4.4	4.4	15.3	-6.7	-7.1
	Commercial	1986-1992	3.8	6.3	10.7	6.5	-8.9	-15.1
Garland	Residential ^c	1988-1992	— ^d	3.6 ^e	3.8	— ^d	6.1 ^e	6.6
	Residential ^f	1988-1992	— ^d	3.0 ^e	3.0	— ^d	5.2 ^e	4.9
	Gen. Service	1988-1992	— ^d	9.1 ^e	6.3	— ^d	13.4 ^e	6.2
	Public Sector	1988-1992	— ^d	4.1 ^e	2.9	— ^d	9.1 ^e	5.3
Greenville	Residential	1994-1998	— ^g	— ^g	— ^g	— ^g	— ^g	— ^g
	Gen. service	1994-1998	— ^g	— ^g	— ^g	— ^g	— ^g	— ^g
	Total	1989-1993	8.2 ^e	2.7	3.9	12.0 ^e	3.3	4.5

^b Forecasting model needed to generate unconditional forecasts not supplied by utility staff.

^a City

^b Rural

^c Customers using gas.

^d Forecasting model needed to generate conditional forecasts not supplied by the utility staff.

^e Medium forecast.

^f Customers using electricity.

^g Not available; forecast period not appropriate for error calculations.

Table 4.35 Summary of Conditional Total System Peak Load Demand Forecasts.

City	Forecast Period	Average (MSE) Annual Forecast Error (%)			Maximum Annual Forecast Error (%)		
		Utilit	NN-	NN-	Utilit	NN-	NN-
		y Staff	1	2	y Staff	1	2
Bryan	1984-1990	18.1	11.8	6.4	29.0	20.7	12.4
Denton	1986-1992	10.2	4.8	7.4	20.3	8.5	11.5
Garland	1988-1992	— ^a	3.4 ^b	4.8	— ^a	5.9 ^b	7.7
Greenville	1989-1993	26.5 ^b	6.0	8.9	40.3 ^b	11.2	14.5

1. The neural network forecasting models consistently exhibit improved accuracy in terms of average annual forecast error and maximum forecast error for energy sales as well as peak load demand compared to utility conducted forecasts. The only exception being the commercial class for the City of Denton.
2. With the exception of the commercial customer classes of the City of Bryan, the NN-1 model average forecast errors for energy sales are less than 9.1% and the maximum NN-1 forecast errors for energy sales are less than 14.5%. With the exception of the commercial customer classes of City of Bryan, the NN-2

^a Forecasting model needed to generate conditional forecasts not supplied by the utility staff.

^b Medium forecast.

model average forecast errors for energy sales are less than 10.7%, and the maximum forecast errors for energy sales are less than 15.1%.

3. The NN-1 model produced the lowest average annual forecast errors in peak load demand for each of the cases studied. However, the NN-2 model produced the lowest maximum forecast error in peak load demand for 2 of the 4 cases studied. No unconditional forecasts of peak load demand were available from the participating utilities.
4. The NN-1 model average forecast errors for peak load demand are less than 11.8%, and the maximum forecast errors for peak load demand are less than 20.7%. The NN-2 model average forecast errors for peak load demand are less than 8.9% and the maximum forecast errors for peak load demand are less than 14.5%. Both neural network models provide much better accuracy than utility models.
5. The NN-2 neural network model gave better accuracy in forecasting energy sales both in terms of average annual error and maximum error than the NN-1 model in a majority of the cases studied.

A characteristic of the NN forecasting models is their observed robustness. Such behavior has also been reported in short-term load forecasting studies using neural networks [49]. A look at the Tables 4.32 and 4.34 reveals that the effect of exogenous inputs uncertainty to the forecast error is more pronounced in the utility supplied forecasts as compared to the NN forecasts. This is an indication of neural network model robustness and it could be attributed to the piece-wise linear fitting properties of neural networks.

IV.6 Chapter Summary

In this chapter, methods developed in chapter III for long-term load forecasting were applied to energy sales in member utilities of Texas Municipally Power Agency (TMPA). These members include City of Bryan, City of Denton, City of Garland and City of Greenville. The energy forecast for different classes of customers as well as the total system peak load and system demand was performed.

The member utilities had provided the required historical data and other crucial expertise to perform such long-term load forecasts, and generate energy sales and peak load demand forecasts. For utilities, which had provided a reference forecast performed in past, an attempt was made to use only the amount of information that was available to the forecaster at the time the reference forecast had been developed. For utilities which had not provided any reference forecast, a number of years were excluded from the recent past and the energy sales and peak load demand was forecasted for the excluded years using the remaining historical observations. For the city of Greenville, due to extremely short historical observations, no comparison with other historical time series were performed.

Details of unconditional and conditional forecast of energy sales in different classes of customers for the City of Bryan as well as the peak load and total system demand was presented. Two neural network forecasting methods were used in developing the forecasts. The forecasts were compared with the available forecasts from the City of Bryan Utility Staff. To perform conditional forecasts, the projection of different exogenous variables available at the time of the reference forecast were used.

Both neural network forecasting methods outperformed the available unconditional forecasts from the City of Bryan. The improvement of NN-1 forecasting

method compared to the available forecasts ranged from 10% to 49% on the average forecast error and from 3% to 37% on the maximum annual forecast error. The improvement of NN-2 forecasting method compared to the available forecasts ranged from 18% to 71% on the average forecast error and from 15% to 64% on the maximum annual forecast error.

Other utilities did not provide any forecasts to serve as the reference forecast in this study. The average unconditional forecast error for different customer classes of other utilities for NN-1 and NN-2 forecasting methods were below 10% and 7.7%, respectively and the maximum annual unconditional forecast error for NN-1 and NN-2 forecasting methods were below 14% and 11.3% respectively. The average conditional forecast error for different customer classes of other utilities for NN-1 and NN-2 forecasting methods were below 9.1% and 10.7% respectively, and the maximum annual conditional forecast error for NN-1 and NN-2 forecasting methods were below 14.5% and 15.1%, respectively.

UNCERTAINTY QUANTIFICATION: RESAMPLING TECHNIQUES

V.1 Introduction

The basic question in statistics is to infer statements about a distribution of infinite samples, given a finite sample set at hand. Traditionally, this problem has been attacked using assumptions about the nature of the original unknown distribution. Such initial assumptions confine the problem within a parametric framework. Moreover, such assumptions enable the statistician to develop analytical solutions to problems which otherwise seemed unsolvable in a nonparametric context.

A wave of advances in the statistical theory has emerged, with the introduction of computers to the statistics community in late 1970's [24]. Using computers has relieved the statistician from adopting unrealistic assumptions about the nature of the unknown generating distribution function. However, the price to pay is intensive use of computers. Among these techniques are the general resampling techniques, robust estimation and the projection pursuit method.

This chapter is organized as follows. Section V.2 briefly defines the resampling methods. Section V.3 introduces bootstrapping method and its application in assessing the variability of an estimate statistic. As an example, the application of the bootstrap method to measure the standard error of the average mean is discussed. Discussion of the application of the bootstrap method is accompanied with explanations of several important concepts in bootstrap application such as: random selection with replacement and bootstrap sample size. Section V.4 presents several nonparametric confidence

interval computation methods and discusses the advantages and drawbacks of each method. Section V.5 presents a discussion on the effect of a particular point forecaster on the computed confidence bands of the energy demand. Section V.6 summarizes the chapter.

V.2 Resampling Methods

Resampling methods refer to methods in which the observed data are sampled repeatedly, in a computer intensive simulation analysis, to provide inferences [73]. In resampling, the observed values are randomly reassigned to treatment groups, and the test statistics are recomputed. The assignment and recomputations are done hundreds of times.

Given a set of numbers, their average can be a representation of the average of the original population which includes the sample set. However, in a resampling context, the sample set elements can be randomly resampled repeatedly, and pseudo data sets be generated. Since the pseudo data set is randomly selected, there should be no difference between the averages of such sets beyond that which can occur by chance. Computing the average of the pseudo data sets repeatedly a large number of times enables one to compute the mean of the means of the pseudo data sets and thereby to infer the mean of the original population.

Using resampling methods enables the statistician to analyze a large [38], [73] class of statistical problems. Among resampling techniques are parametric and nonparametric bootstrap methods, permutation analysis, Jackknife and data-splitting methods.

V.3 Bootstrapping Techniques

A typical problem in statistics involves the estimation of an unknown parameter θ . The two main questions asked are : (1) What estimator $\hat{\theta}$ should be used, and (2) Having chosen to use a particular $\hat{\theta}$, how accurate is it as an estimator of θ . The bootstrap method, which was formulated by Efron 1979 tries to answer the second question [24], [25].

Assume that an observed data set X has been generated from an unknown distribution F as:

$$X_1, X_2, \dots, X_n \sim F \quad (5.1)$$

Having observed $X_1 = x_1, X_2 = x_2, \dots, X_n = x_n$, the sample mean is $\bar{x} = \sum_{i=1}^n x_i / n$.

The question is how accurate is \bar{x} as an estimate of the true mean $\theta = E\{X\}$. An estimate of the accuracy of \bar{x} is the well known standard error equation:

$$\bar{\sigma} = \left[\frac{1}{n(n-1)} \sum_{i=1}^n (x_i - \bar{x})^2 \right]^{1/2} \quad (5.2)$$

where $\bar{\sigma}$ is the standard error of $\bar{X} = \bar{x}$. The problem with equation (5.2) is that it can not be extended to other statistics, such as the median [26]. The Jackknife and bootstrap methods are two ways for making this extension. In the following, the standard error of an estimate will be computed using the Jackknife and the bootstrap methods and compared with the normal theory estimate.

Deleting the i th point from the current sample set, the resulting sample mean is computed as:

$$\bar{x}_{(i)} = \frac{n\bar{x} - x_i}{n-1} = \frac{1}{n-1} \sum_{j \neq i} x_j \quad (5.3)$$

and the average of the means of such samples can be expressed as:

$$\bar{x}_{(\bullet)} = \sum_{i=1}^n \bar{x}_{(i)} / n \quad (5.4)$$

Then, the Jackknife estimate of the standard error will be:

$$\hat{\sigma}_j = \left[\frac{n-1}{n} \sum_{i=1}^n (\bar{x}_{(i)} - \bar{x}_{(\bullet)})^2 \right]^{1/2} \quad (5.5)$$

The advantage of (5.5) is that it can be generalized to any other statistical estimate of interest: $\hat{\theta}_{(i)} = \hat{\theta}(X_1, X_2, \dots, X_n)$. The only change is to substitute $\hat{\theta}_{(i)} = \hat{\theta}(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n)$ for $\bar{x}_{(i)}$ and $\hat{\theta}_{(\bullet)} = \sum_{i=1}^n \hat{\theta}_{(i)} / n$ for $\bar{x}_{(\bullet)}$.

The bootstrap generalizes (5.2) in a different way. Let \hat{F} be the empirical probability distribution of the data, putting probability weight $1/n$ on each $x_{(i)}$, and let $X_1^*, X_2^*, \dots, X_n^*$ be a random sample of \hat{F} :

$$X_1^*, X_2^*, \dots, X_n^* \sim \hat{F} \quad (5.6)$$

In other words, each X_i^* is drawn independently with replacement and with equal probability from the set $\{x_1, x_2, \dots, x_n\}$. Then, $\bar{X}^* = \sum_{i=1}^n X_i^* / n$ has the variance:

$$\text{Var}_* \bar{X}^* = \frac{1}{n^2} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (5.7)$$

where Var_* identifies the variance under sampling scheme of (5.6). The bootstrap estimate of the standard error for an estimator $\hat{\theta} = \hat{\theta}(X_1, X_2, \dots, X_n)$ will be given as:

$$\hat{\sigma}_B = [\text{Var}_* \hat{\theta}(X_1^*, X_2^*, \dots, X_n^*)]^{1/2} \quad (5.8)$$

Figure 5.1 is a schematic illustration of the bootstrap method for a general probability model F . Using the only available realization of F at hand, X , the statistic of interest $\hat{\theta}$

is estimated. Using the X , an estimate of F is obtained as \hat{F} through a Monte Carlo type operation. Realizations X^* of \hat{F} are used to develop an ensemble of estimates $\hat{\theta}^*$.

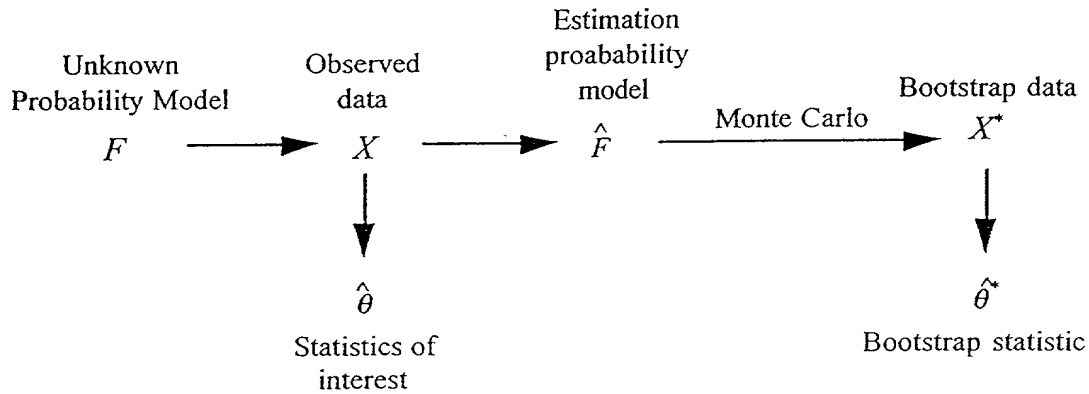


Figure 5.1: Schematic Illustration of Bootstrap [27].

Using this ensemble, the variability of $\hat{\theta}^*$ in \hat{F} is estimated as an estimate of the variability of $\hat{\theta}$ in F . An underlying assumption in general statistical equations is the normality of the unknown distribution function F . However, for finite sample sets, and especially when the number of samples is small, this assumption may not be valid. Using the empirical distribution of the data set in hand to generate the bootstrap sample set \hat{F} relieves one from the normality assumption. However, this is at the expense of intensive computational effort.

Now, consider an example given in Efron's introduction to bootstrap. Table 5.1 and Figure 5.2 show the 1973 entering classes of 15 American law schools [24], [25]. For each school two numbers are given:

x_i = average LSAT score of entering students in the Law school i .

863

y_i = average GPA of entering students in the Law school i .

The correlation coefficient is a measure of association between two sets of infinitely large set of numbers. By definition, the correlation coefficient is written as:

[illegible]

Table 5.1: The Average LAST and GPA at 15 American Law Schools, at 1973 [24], [25].

School #	1	2	3	4	5	6	7	8
LSAT	576	635	558	578	666	580	555	661
GPA	3.39	3.30	2.81	3.03	3.44	3.07	3.00	3.43
School #	9	10	11	12	13	14	15	
LSAT	651	605	653	575	545	572	594	
GPA	3.36	3.13	3.12	2.74	2.76	2.88	2.96	

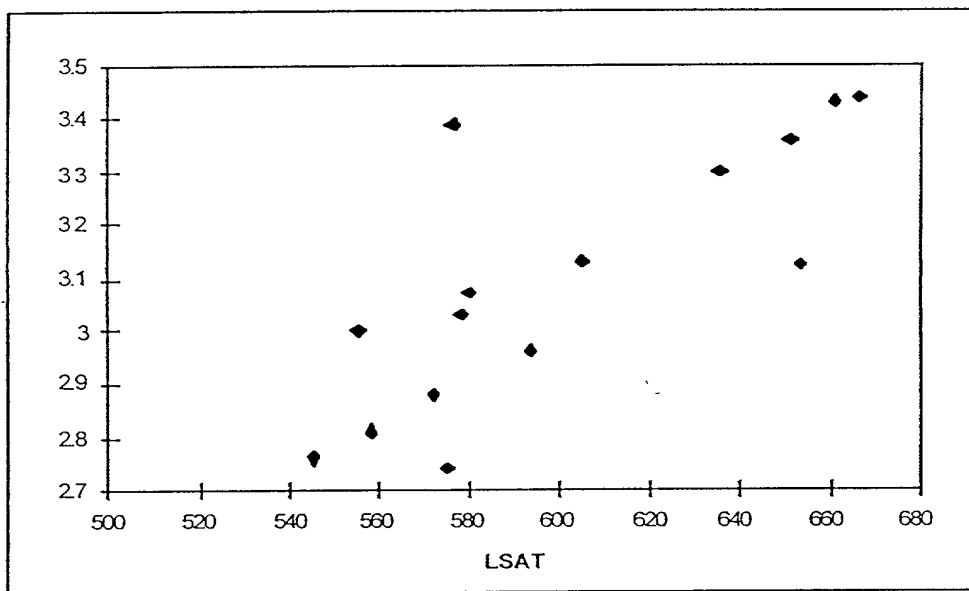


Figure 5.2: A Plot of Law School Data Given in Table 5.1 [24], [25].

$$\hat{\rho} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (5.9)$$

where,

$$\bar{x} = \sum_{i=1}^n x_i / n, \quad \bar{y} = \sum_{i=1}^n y_i / n \quad (5.10)$$

For the above set of LSAT and GPA data set, $\hat{\rho} = 0.776$. This shows a strong positive correlation between the LSAT and GPA. The most common measure of accuracy of $\hat{\rho}$ is the standard deviation:

$$\sigma = \sqrt{E(\hat{\rho} - \rho)^2} \quad (5.11)$$

based on a normality assumption and infinite data points. Accurate assessment of the estimated correlation coefficient is based on the assumption that the available samples have been drawn from a bivariate normal distribution with correlation coefficient $\hat{\rho}$. The exact density function of ρ can be computed theoretically. This density function depends only on $\hat{\rho}$, and not on means or the standard deviation of x and y . This density function can be defined as:

$$\int_a^b f_{\rho}(\hat{\rho}) d\hat{\rho} = \text{Prob} \{a \leq \hat{\rho} \leq b\} \quad (5.12)$$

Efron and his colleagues [24], [26], [27] have reported the results of the above formulation.

Figure 5.3 shows $f_{\rho}(\cdot)$ for $\hat{\rho} = 0.776$. The abscissa is plotted on $\hat{\rho}^* - \hat{\rho}$. Clearly, the density function is not exactly normal, nor symmetric. The normal theory estimate of σ , $\hat{\sigma}^{(N)}$, is half the length of the central 68% population of the distribution described by $f_{\hat{\rho}}(\cdot)$. That is, $\hat{\sigma}^{(N)}$ is half of the interval between the 16th percentile and 84th percentile. For $\hat{\rho} = 0.776$, this estimate is $\hat{\sigma}^{(N)} = 0.113$.

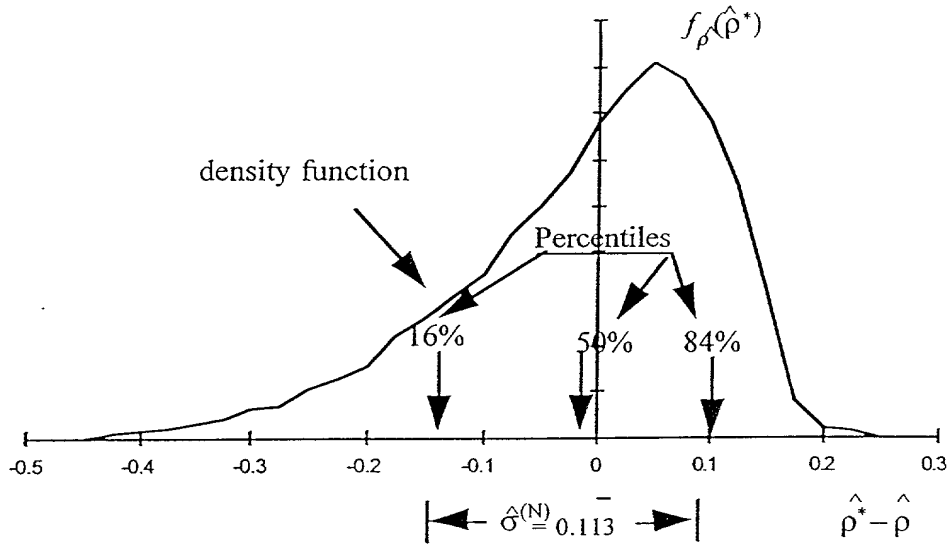


Figure 5.3: The Normal Theory Density Function Drawn From a Bivariate Normal Distribution With True Correlation $\hat{\rho}=0.776$ [24], [25].

Now, obtain the Jackknife estimate of σ as follows. The pair (x_i, y_i) is deleted from the data set, and the correlation coefficient is recomputed for the remaining 14 points. Using this method, 15 new $\hat{\rho}$ will be computed: $\hat{\rho}(i)$, $i=1, \dots, n=15$. Therefore, Jackknife estimate of the standard error is given by:

$$\hat{\sigma}^{(j)} = \sqrt{\frac{(n-1)}{n} \sum_{i=1}^n (\hat{\rho}(i) - \bar{\rho}(\bullet))^2}, \quad \bar{\rho}(\bullet) = \frac{1}{n} \sum_{i=1}^n \hat{\rho}(i) \quad (5.13)$$

$$\hat{\sigma}^{(j)} = 0.143 \quad (5.14)$$

To apply the bootstrap method to the current problem, a Monte Carlo algorithm should be performed as given in the following steps:

1. Let \hat{F} be the empirical distribution of the 15 observed data points.

2. Randomly select with replacement from \hat{F} , to generate bootstrap sets of 15 data points. Some of the original pairs will be selected more than once, some none.
3. Compute $\hat{\rho}^*$, for each bootstrap sample set.
4. Repeat the Monte Carlo process for a large number of times N.

The bootstrap estimate of the sample standard error will be:

$$\hat{\sigma}^{(B)} = \left(\frac{\sum_{i=1}^N (\hat{\rho}^*(i) - \bar{\rho}^*(\bullet))^2}{N-1} \right)^{1/2} \quad (5.15)$$

where

$$\bar{\rho}^*(\bullet) = \frac{\sum_{i=1}^N \hat{\rho}^*(i)}{N} \quad (5.16)$$

Having the bootstrap histogram of $\hat{\rho}^*$ obtained through the Monte Carlo process, it is also possible to compute the bootstrap estimate of the standard error using the central 68% interval of $\hat{\rho}^*$. Half the length of the central 68% of the population is an estimate of the standard error, therefore:

$$\frac{\#\{\hat{\rho}^*(i) < a^*\}}{N} = 0.16, \quad \frac{\#\{\hat{\rho}^*(i) < b^*\}}{N} = 0.84 \quad (5.17)$$

Then,

$$\hat{\sigma}^{(B)} = \frac{b^* - a^*}{2} \quad (5.18)$$

This gives $\hat{\sigma}^{(B)} = 0.128$ for the current problem. Figure 5.4 shows a histogram of N=1000 bootstrap replications of $\hat{\rho}^*$ for the law school data. Also shown are the 16% and 84% percentiles accumulating $2\hat{\sigma}^{(B)}$ mass of the bootstrap estimate of $\hat{\rho}^* - \hat{\rho}$.

Table 5.2 shows the change in the estimate of standard error of $\hat{\rho}^* - \hat{\rho}$ using equations (5.15) and (5.18) as the number of bootstrap replications changes. As N

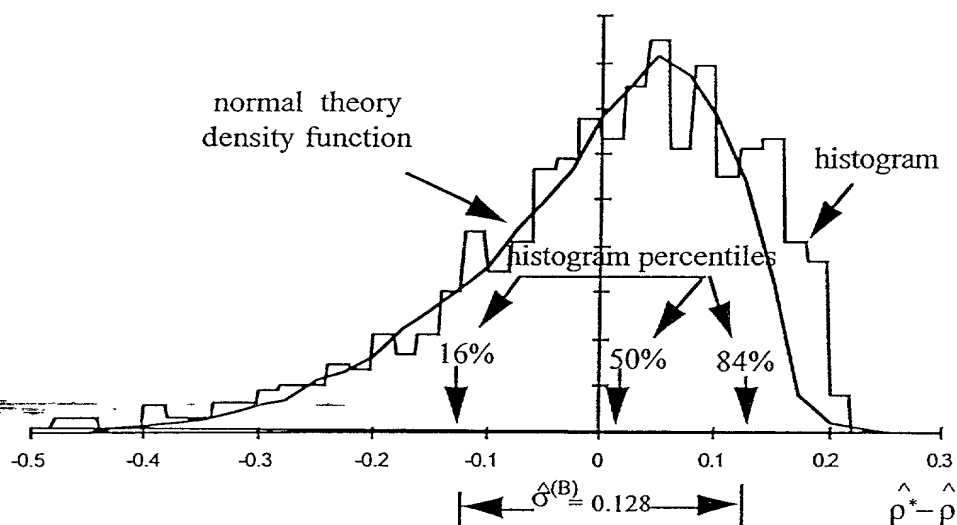


Figure 5.4: Histogram of $N=1000$ Bootstrap Replications of $\hat{\rho}^*$ for the Law School Data.

Table 5.2: Bootstrap Estimate of Standard Error for Different Replications.

Bootstrap Replications	Bootstrap estimate of Standard Error	
	Equation (5.15)	Equation (5.18)
50	0.106	0.125
200	0.123	0.124
500	0.132	0.122
1000	0.132	0.128
5000	0.130	0.131
10000	0.129	0.131
30000	0.130	0.131

increases, the change in $\hat{\sigma}^{(B)}$ diminishes. However, the price is more computational effort.

The above discussion serves as an introduction to the subject of resampling and bootstrapping techniques. The literature on the application of bootstrapping techniques is substantial. Freedman [30] reported the first application to regression modeling. Freedman and Peters [31] applied the bootstrap method to assess the variability of the model parameters in a set of simultaneous regression equations modeling the energy demand in 10 different U.S. industrial regions. They concluded that bootstrap results in more accurate estimates of the variability of the model parameters. Moreover, they [32] forecasted the energy demand in the aforementioned regions taking into account the variability in the model parameters using bootstrap. Chatfield [17] reported a comparison between analytical, approximate analytical, bayesian, and bootstrap techniques to compute prediction intervals and concluded that bootstrap results in more accurate estimates of the prediction intervals. Efron [24] proposed several methods to compute the bootstrap confidence intervals. Buckland [14], [15] reported and compared four methods to compute bootstrap confidence intervals. Hutsch [38] formulated bootstrap for general stationary observations.

Two important concepts in bootstrap application are explained in the following two subsections.

V.3.1 Resampling With Replacement

In bootstrap resampling, one creates random variables having distribution \hat{F} . Let $X_1^*, X_2^*, \dots, X_n^*$ each be generated randomly and independently as follows:

$$X_i^* = \begin{cases} x_1, & \text{with probability } \frac{1}{n} \\ x_2, & \text{with probability } \frac{1}{n} \\ \vdots & \\ x_n, & \text{with probability } \frac{1}{n} \end{cases} \quad (5.19)$$

Then, the distribution of X_i^* is the empirical distribution function \hat{F} [73]. A typical random sample from this distribution may be obtained by randomly sampling x_1, \dots, x_n with replacement. Let x_1^* be a random selection of the observed data set x_1, \dots, x_n , say x_j . Next x_j is returned to the collection, and x_2^* is sampled from the same collection from which x_1^* was sampled. This process continues until all values through x_n^* have been selected. At each step, the value selected is returned to the pool of data. The term *with replacement* refers to sample sets constructed in this way. It should be noted that some of the original x_j data values are likely to appear more than once among the sampled values x_j^* , while others will not appear at all. On the other hand, in samples *without replacement*, the values are not returned to the pool of data after having been selected. In this case, each original value x_j appears once and only once among sampled values x_j^* . This method is not useful in generating bootstrap samples, since the x_j^* are constant regardless of the order of the data values.

V.3.2 Bootstrap Sample Size

The bootstrap method tries to generate \hat{F} as close as possible to F by random selection with replacement from the only available realization of F , X . As long as new

bootstrap replications cause a noticeable change in \hat{F} , the random sampling process continues.

In the law school data example (Figure 5.2 and Table 5.1), the variation in the standard error diminishes after 1000 iterations. In fact, $N=200$, provides an acceptable estimate.

In general, the number of sampling repetitions required depends on the nature of F , which is the unknown generating distribution of the statistics of interest. Moreover, it is obvious that the confidence interval of the forecast is a more ambitious statistics of interest than the standard error. Therefore, while 200 may be an acceptable number of repetitions for standard error, it is definitely not acceptable in estimating the confidence interval of forecasts.

V.4 Nonparametric Bootstrap Confidence Interval Computation

The bootstrap method is a nonparametric statistical technique. It generates an ensemble of estimates. The assessment of the variability of the estimate is transformed to assessment of the variability of the estimate in the bootstrap generated ensemble. Efron [24] proposed several methods to measure the variability of the estimate in the bootstrap generated ensemble. Buckman [14], [15] reported four different methods to compute confidence intervals (CI) in parametric and nonparametric settings. Mooney and Duval [47] discussed other CI computation techniques. The selection of a particular method depends on the nature of the problem at hand. While assuming a parametric setting removes the necessity of further computations, some nonparametric techniques require an individual bootstrap ensemble to be generated around each estimate of the original bootstrap estimates. In the following subsections several CI computation techniques will be reviewed.

V.4.1 Normal Approximation Method

The normal approximation method is quite analogous to the parametric approach for constructing CIs. If it is reasonable that the parameter statistic is normally distributed, but no analytical standard error formula exists, the standard error can be calculated as:

$$\hat{\sigma}_{(B)} = \left(\frac{\sum_{i=1}^N (\hat{\theta}^*_{(i)} - \bar{\theta}^*_{(\cdot)})^2}{N-1} \right)^{1/2} \quad \bar{\theta}^*_{(\cdot)} = \frac{\sum_{i=1}^N \hat{\theta}^*_{(i)}}{N} \quad (5.20)$$

The above standard errors are used to transform the data to Z or student's t distribution with $\alpha/2$ and $1-\alpha/2$ probability limits. Then, similar to the traditional CI formula:

$$P(\hat{\theta} - z_{\alpha/2} \hat{\sigma}_{\hat{\theta}}^* < \theta < \hat{\theta} + z_{\alpha/2} \hat{\sigma}_{\hat{\theta}}^*) = 1 - \alpha \quad (5.21)$$

The results of this approach are very close to those obtained using analytical techniques. The advantage is simplicity. If the normality assumption holds, this method generates the most accurate estimate of the confidence interval. However, if the normality assumption is not fulfilled, the accuracy of this method is doubtful.

V.4.2 Symmetric Percentile Method

The symmetric percentile method uses the notion that $\hat{F}(\hat{\theta}^*)$ accurately approximates $F(\hat{\theta})$, the original distribution. The basic approach is that an α -level CI includes all the values of $\hat{\theta}^*$ between the $\alpha/2$ and $1-\alpha/2$ percentiles of the $\hat{F}(\hat{\theta}^*)$ distribution. Thus, the end points for a 0.05α -level confidence interval for $\hat{\theta}$ would be $\hat{\theta}^*$ at the 2.5th and 97.5th percentiles of $\hat{F}(\hat{\theta}^*)$. In theory, the percentile method allows $\hat{F}(\hat{\theta}^*)$ to conform to any shape that the data suggests. This allows the CI to be

asymmetric around the expected value of $\hat{\theta}$. The percentile method also has the advantage of being simple to apply. No complex analytical formula is required to estimate $\hat{\theta}$'s. One only needs to calculate the $\hat{\theta}^*$'s, sort them, and these count up and down. For these reasons, it is the most widely used nonparametric bootstrap CI computation technique.

The percentile method has two drawbacks. It may perform poorly with small samples primarily because of the importance of the tails of the sampling distribution in these CI calculations. Moreover, one must assume that the bootstrapping sampling distribution is an unbiased estimate of $F(\hat{\theta})$. This is a less restrictive assumption, but it may cause concern.

V.4.3 Bias Corrected Percentile Method (BC)

The bias corrected or BC method was suggested as an improvement to the percentile method to overcome the assumption of unbiasedness of $\hat{F}(\hat{\theta}^*)$ [24], [15]. In the bias corrected method, instead of requiring that $\hat{\theta}^* - \hat{\theta}$, and $\hat{\theta} - \theta$ be centered at zero (that is $\hat{\theta}^*$ and $\hat{\theta}$ are unbiased estimators of $\hat{\theta}$ and θ respectively), the BC method assumes that these quantities are distributed around a constant $z_o\sigma$ where σ is the standard deviation of the distribution. The quantity z_o is a biasing constant for which we need to adjust the bootstrap distribution of $\hat{\theta}^*$.

To make the adjustment, the BC method assumes that there is a monotonic transformation function of $\hat{\theta}$ and θ , say $\hat{\phi}$ and ϕ whose differences are normally distributed as:

$$\hat{\phi} - \phi \sim N(z_o\sigma, \sigma^2), \text{ and } \hat{\phi}^* - \hat{\phi} \sim N(z_o\sigma, \sigma^2) \quad (5.22)$$

It should be noted that we do not need to know the specific transformation function, only that such a transformation exists. Assuming the possibility of a normal distribution, we can calculate the value of z . As a result, BC adjusts the upper and lower endpoints of the bootstrap empirical distribution unequally since the relationship between the monotonic z and its normal distribution is nonlinear.

The main problem with the BC method is the parametric assumption. First, we must assume that there must exist some monotonic transformation of $\hat{\theta}$ and θ whose difference has a known distribution such as the normal distribution. The second problem is the assumption of unbiasedness of the original estimator $\hat{\theta}$.

V.4.4 Minimum Length Method

For a certain confidence level, the bands of the required population mass can be measured based on different criteria such as symmetric around the mean, or the median. Buckland [14], [15] proposed the selection of the minimum length which bounds the required percentage of the total population as a measure of the confidence bands.

Mathematically, for 95% upper and lower confidence bands, the minimum length method computes on i such that the difference between the i th smallest and the $(95+i)$ th largest estimate is the minimum length ($i=1, \dots, 5$). This estimate i is as an approximate lower and upper 95% confidence limits.

This method is effected by sampling variations although this source of error will be relatively small when several hundred estimates are used in CI estimation.

V.5 Effect of Point forecaster on Forecast Uncertainty

Uncertainty assessment in a forecasting model should account for both model parameter and input parameters (conditional) uncertainty. The bootstrapping method is used to assess model uncertainty. Conditional uncertainty is assessed by randomly drawing from future distribution of the exogenous variables. Each future realization of exogenous variables is passed through the current model and an ensemble of forecasts is generated based on each bootstrap model. The variability in the resulting ensemble of the energy demand forecasts, presented in the form of confidence bands, is an assessment of both model parameter and input parameter uncertainties.

The confidence bands in the generated forecast ensemble can be computed using a nonparametric CI computation method. Nonparametric methods require a minimum number of ensemble realizations for a particular confidence level. Obviously, as the required confidence level increases, the required number of ensemble realizations increases. In long term load forecasting practice, it is common to use 95% confidence bands.

The nonparametric CI computation method must converge if the model uncertainty for a particular confidence level is to be assessed reliably. Similarly, to reliably assess the conditional uncertainty for a particular confidence level, the nonparametric CI computation method must converge. Therefore, to assess the total uncertainty, the number of realizations is multiplication of the two numbers. Experience has shown that for a confidence level of 95% the required number of bootstrap realizations is at least 500. The number of realizations required for accurate results depends on the type of estimator, the variability of the model parameters and the distributions of the exogenous variables.

In the previous section several CI methods were presented and compared. The minimum length method has been chosen to calculate CIs of load forecasts. The computational requirement of the minimum length method is not significantly higher than other CI methods and the method does not have the drawbacks discussed for the other CI computation methods.

In this section the effect of the particular point forecasting method on forecast uncertainty is discussed. The different forecast estimators discussed in this section are ordinary least squares (OLS) estimator which represents the widely used regression models, the regularized linear regression, and regularized neural network estimators. It must be clearly stated that the purpose of this comparison is not to compare the point forecasting accuracy of different estimators. This study investigates the effect of the point forecaster on the computed confidence bands. Chapter IV showed that regularized neural network models outperformed regression models in 85% of the forecasts developed for TMPA. The regression model developed for this study has been generated using statistical packages, and had not been provided by the City of Bryan.

Energy sales for the City of Bryan city residential class is forecasted and the associated uncertainty is computed using each of the above estimators in a bootstrapping frame work. This class of customers was introduced in section IV.4. The energy sales model includes four exogenous variables: time, number of residential customers, CDDs, and the price of electricity.

Now, consider the uncertainties of the exogenous variables for the City of Bryan city residential class. The time variable obviously does not include any uncertainty. The general practice in reporting projections of future customers is to specify the Hi, Medium and Low growth rates. Through consultation with the City of Bryan utility staff, customer growth projections were set to 4%, 2.5% and 1.5% average annual

growth rates based on 1987 residential customers, respectively. This projection assumes the medium forecast is the most probable future scenario. It was decided to represent such an uncertainty as a normal distribution with Hi, Medium and Low projections being 97.5%, 50% and 2.5% probability levels. Figure 5.5 shows the historical number of customers time series as well as the Medium forecast with a 95% confidence bands. Table 5.3 shows the mean forecast, forecast errors and upper and lower 95% confidence limits on the forecast.

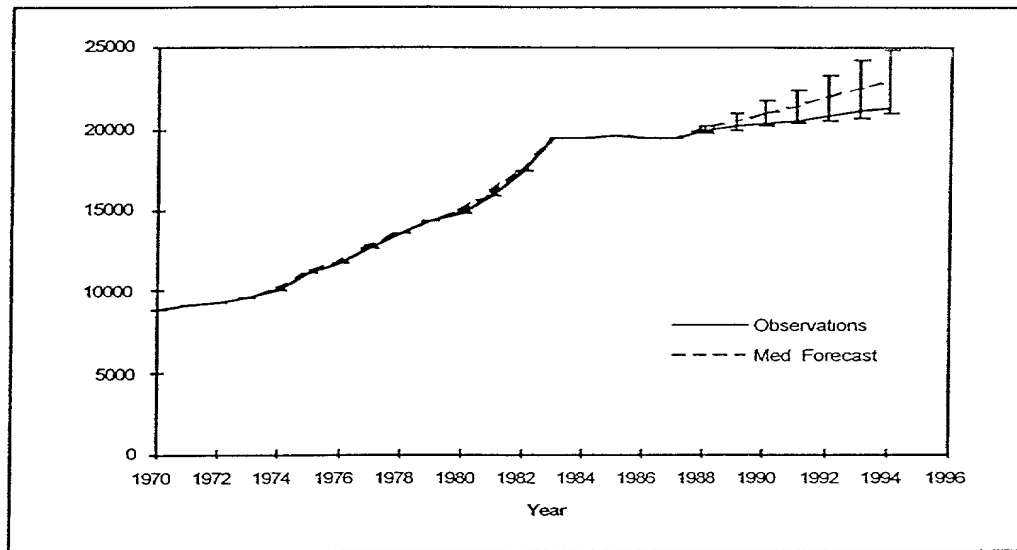


Figure 5.5: Forecast of the Residential Customer and 95% Confidence Bands.

Figure 5.6 and Table 5.4 show the historical CDDs in the service area. Due to unavailability of any forecast of the CDDs in the service area, it was decided to forecast the CDDs in the service area as a time series with its value being the average of the 18 years of the historical CDDs.

Similar to the projection of the residential customers, the general practice in reporting future values of the price of electricity levels is to specify the Hi, Medium, and Low projections of future electricity price in cents per KWh. These projections for the City

of Bryan city residential class are 1.85%, 2.9% and 4.45% respectively based on the 1987 price of electricity in cents per Kwh. These projections set the medium forecast to be the most probable future scenario. Through discussion with the City of Bryan utility staff, it was decided to present these projections as a normal distribution with Hi, Medium, and Low forecasts being 97.5%, 50%, and 2.5% probability levels respectively.

Figure 5.7 and Table 5.5 depict the historical electricity price time series, the Medium forecast, forecast errors and upper (97.5%) and lower (2.5%) bands. It is

Table 5.3: Medium Forecast and 95% Confidence Bands of City Residential Customers.

Fiscal Year	Customers	Forecast and Forecast Error of Number of Customers			
1970	8876				
1971	9066				
1972	9378				
1973	9570				
1974	10138				
1975	11242				
1976	11790				
1977	12672				
1978	13606				
1979	14281				
1980	14942				
1981	15985				
1982	17476				
1983	19340				
1984	19381				
1985	19658				
1986	19577				
1987	19567	Mean forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)
1988	19979	20017	0.2	19781	20253
1989	20214	20477	1.3	19973	20981
1990	20430	20948	2.5	20182	21714
1991	20606	21430	4.0	20392	22468
1992	20803	21923	5.4	20612	23234
1993	21138	22427	6.1	20770	24084
1994	21447	22943	7.0	21036	24850
			4.5		

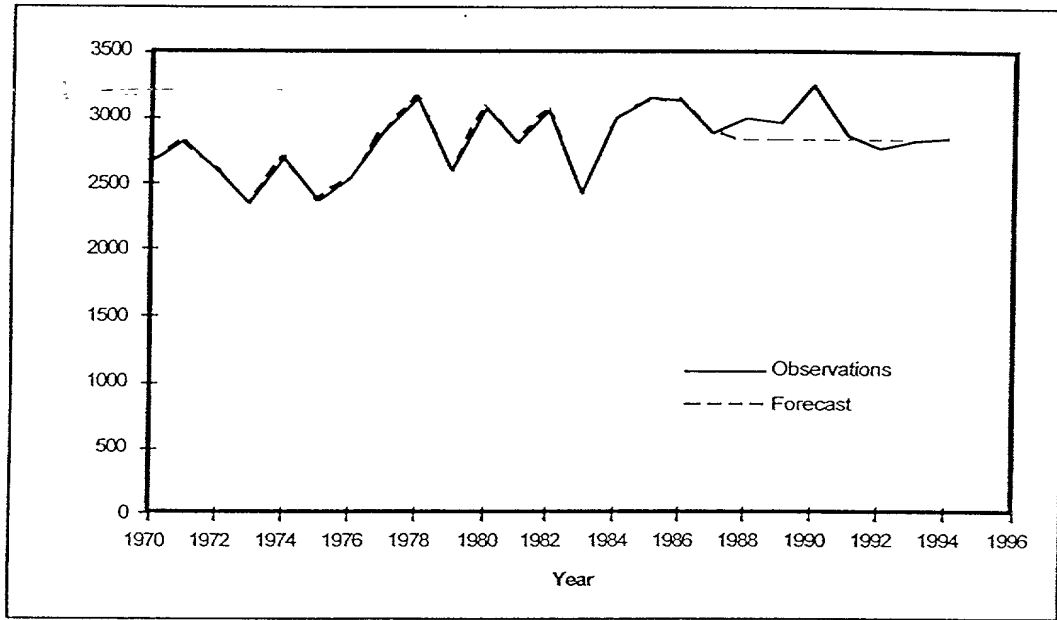


Figure 5.6: Forecast of City of Bryan Cooling Degree Days.

Table 5.4: City CDDs Time Series for the City of Bryan.

Fiscal Year	CDDs	Forecasted CDDs	Error (%)
1970	2696		
1971	2830		
1972	2622		
1973	2363		
1974	2700		
1975	2371		
1976	2553		
1977	2902		
1978	3155		
1979	2626		
1980	3086		
1981	2819		
1982	3074		
1983	2435		
1984	2990		
1985	3156		
1986	3142		
1987	2890		
1988	3004	2838	-5.5
1989	2963	2838	-4.2
1990	3249	2838	-12.7
1991	2873	2838	-1.2
1992	2779	2838	2.1
1993	2825	2838	0.5
1994	2856	2838	-0.6
			6.0

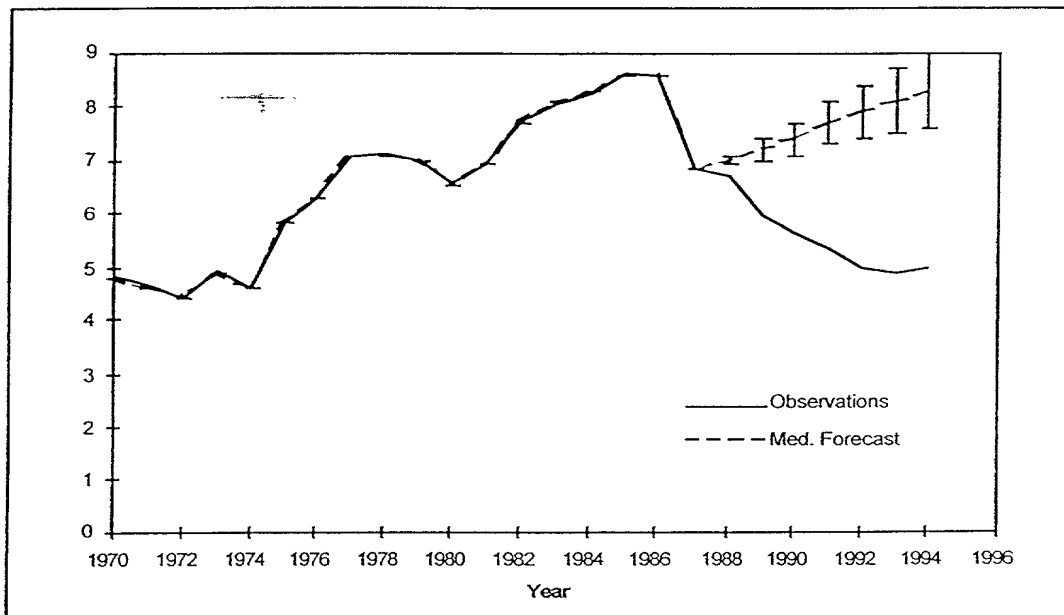


Figure 5.7: Forecast of City Residential Price (cents per Kwh), and 95% Confidence Bands.

Table 5.5: Medium Forecast and 95% Confidence Bands of City Residential Price (c/Kwh).

Fiscal Year	Price (c/Kwh)	Forecast and Forecast Error of Price of Electricity (c/Kwh)			
1970	4.85				
1971	4.62				
1972	4.4				
1973	4.93				
1974	4.58				
1975	5.81				
1976	6.3				
1977	7.1				
1978	7.14				
1979	7.01				
1980	6.53				
1981	6.93				
1982	7.71				
1983	8.05				
1984	8.25				
1985	8.62				
1986	8.55				
1987	6.83	Mean forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)
1988	6.7	7.0	4.5	6.9	7.1
1989	5.97	7.2	20.6	7.0	7.4
1990	5.61	7.4	31.9	7.1	7.7
1991	5.31	7.7	45.0	7.3	8.1
1992	5.00	7.9	58.0	7.4	8.4
1993	4.89	8.1	65.6	7.5	8.7
1994	4.98	8.3	66.7	7.6	9.0
			43.4		

obvious that the price of electricity did not remain within the 95% confidence bands of its forecasted projection.

In the following subsections, the results of application of 3 different point forecasters in assessing the total uncertainty in forecasting the energy sales in the city residential class of the City of Bryan will be analyzed and compared.

V.5.1 Uncertainty Assessment Using OLS Point Forecaster

Ordinary Least Squares (OLS) estimators are the most widely used estimation technique in long-term load forecasting. In its basic form, having historical exogenous indicators and the energy sales time series as X and Y respectively, the forecast model parameters $\hat{\theta}$ are estimated as:

$$\hat{\theta} = \text{inv}(X^T X) X^T Y \quad (5.23)$$

Having the forecast of the exogenous variables in the forecasting period as \hat{X} , a forecast of the energy sales is obtained using the estimated parameters $\hat{\theta}$ and X_f , as:

$$Y_f = X_f \hat{\theta} \quad (5.24)$$

Using the estimated parameters $\hat{\theta}$, the model prediction in the historical period will differ from the observations by a residual vector:

$$\hat{\varepsilon} = Y - \hat{Y} = Y - X\hat{\theta} \quad (5.25)$$

The bootstrapping method utilizes the residual vector to assess the forecasting model uncertainty. Repeatedly drawing with replacement from the residual vector $\hat{\varepsilon}$, bootstrap generates bootstrap residual vectors $\hat{\varepsilon}^*$. Using $\hat{\varepsilon}^*$ and the model prediction in the historical period \hat{Y} , bootstrap generates virtual historical observations Y^* :

$$Y^* = \hat{Y} + \hat{\varepsilon} \quad (5.26)$$

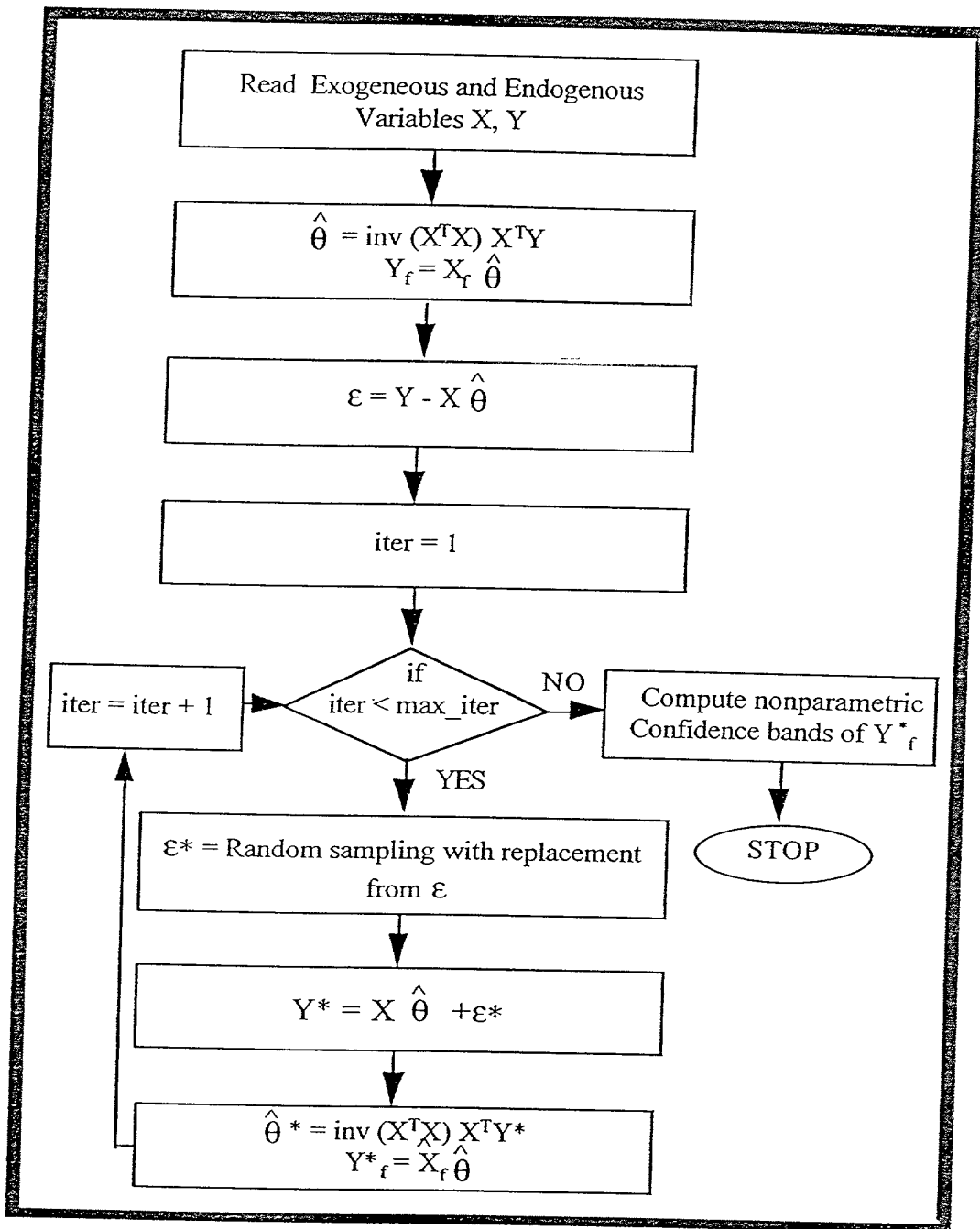


Figure 5.8: Flow Chart of Uncertainty Assessment Using Regression Point Forecaster.

The (X, Y^*) pair is assumed as a historical set, and using an OLS estimator, a bootstrap estimate $\hat{\theta}^*$ is estimated. Using $\hat{\theta}^*$, a new forecast of the future energy sales is obtained. Repeating this process for a large number of iterations, the bootstrap generates an ensemble of forecasts. The forecasting model uncertainty is assessed using the variability in the ensemble of forecasts. If after obtaining each bootstrap model, the future distribution of exogenous variables is used to generate an ensemble of forecasts based on the current bootstrap model parameters, the final ensemble can be used as an assessment of the model parameter as well as indicator uncertainty. The total number of forecast realizations will equal the multiplication of number of bootstrap realizations times the number of randomly selected realizations of the exogenous variables. Figure 5.8 shows a flowchart of uncertainty assessment, using OLS as a point forecaster of energy sales.

Figure 5.9 and Table 5.6 show the forecast of the energy sales in City of Bryan city residential class, and 95% upper and lower confidence bands. Average forecast error is 2.0%. In chapter IV, the conditional forecast of the energy sales in the city residential class using a regression model resulted in 15.9% for the year 1984 to 1990. The large improvements is due to more accurate forecast of the residential customers. Obviously, the uncertainty level represented as the width of the confidence band increases as the forecast moves into future. However, the confidence bands are not symmetric around the point forecast.

Table 5.6 shows the percentage difference of the lower and upper 95% confidence bands compared to the mean forecast. While, in the years 1988 and 1989, the point forecast is closer to the lower band, the point forecast will remain closer to the upper band for the following years. In the year 1988, the width of the 95% confidence

band is 8.3% of the point forecast of the energy sales in the year 1988, and it grows to reach 16.6% of the point forecast of the energy sales in the year 1994.

1988 1989 1990 1991 1992 1993 1994

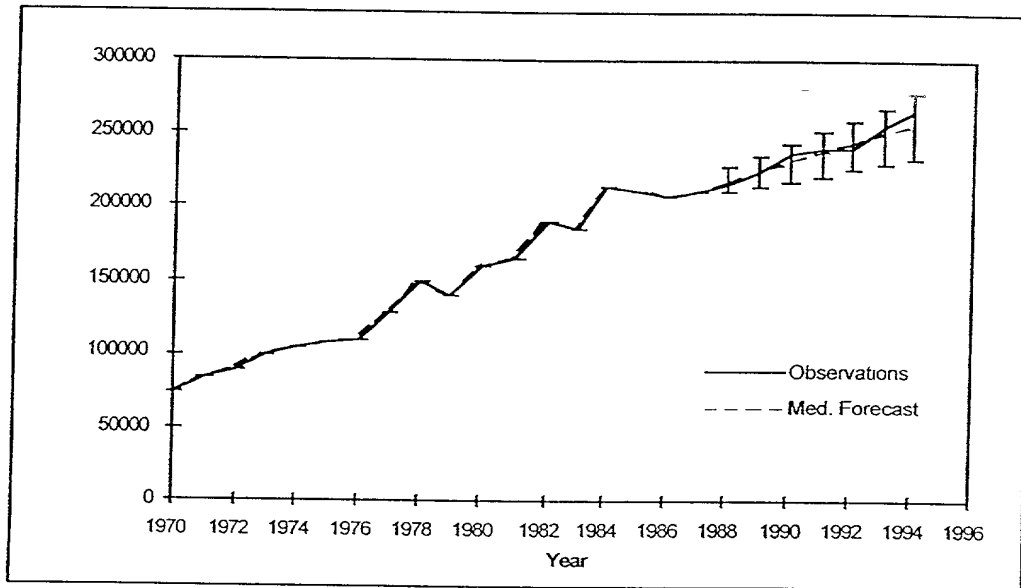


Figure 5.9: Forecast of City Residential Sales and Lower and Upper Limit 95% Confidence Bands in MWH Using Regression Model Point Forecaster.

Table 5.6: Forecast of City Residential Sales , Forecast Error and Lower and Upper Limit 95% Confidence Bands in MWH Using Regression Point Forecaster.

Fiscal Year	Sales (MWH)	Forecast and Forecast Error of Energy sales (MWH)					
1970	74030						
1971	82958						
1972	89745						
1973	99952						
1974	103068						
1975	107629						
1976	109980						
1977	127961						
1978	149597						
1979	138895						
1980	159641						
1981	165912						
1982	191520						
1983	185953						
1984	214618						
1985	212105						
1986	207539	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1987	210360						
1988	216769						
1989	224960						
1990	235767						
1991	240156						
1992	240515						
1993	255000						
1994	266036						
			2.0				

V.5.2 Uncertainty Assessment Using Regularized Linear OLS Point Forecaster

The traditional method of regression analysis is to include all the historical data set in the parameter estimation process directly. However, it is possible to divide the historical data set into two subsets, the training set and the validation set. In the long term load forecasting context, while the training set is directly used in the parameter estimation process, the validation set is used to avoid learning those historical events which will most probably not repeat. Having a training and a validation set, several methods exist to develop forecasting models. These methods have been discussed in detail in chapter III. Among stopping criteria discussed in chapter III, the estimation method which selects models with equal training and testing set errors was adopted in this uncertainty analysis study.

In brief, using a linear gradient descent estimation algorithm, a model is developed using the training set. At every iteration, the performance of the current model parameters on the validation set is evaluated. Models with equal training and validation set performance are selected to forecast the energy demand. Among the top performing models, the model which has the closest forecast to the final average forecast is used for uncertainty analysis. An empirical distribution is generated using the model residual vector. Using the model prediction in the historical period and a bootstrap selection with replacement from the residual vector, a bootstrap set of inputs and outputs will be generated, which once trained will generate a bootstrap forecast of the energy sales. Using the mean forecast of the exogenous variables, the generated ensemble of forecasts will be used to assess the model uncertainty. However, to account for both model and indicator uncertainty, it is possible to pass the future distribution of the exogenous variables through the developed bootstrap models. The final generated forecast ensemble will be an assessment of the

conditional and parameter uncertainty. Figure 5.10 shows a flow chart of the application of the regularized linear regression in uncertainty assessment.

Figure 5.10

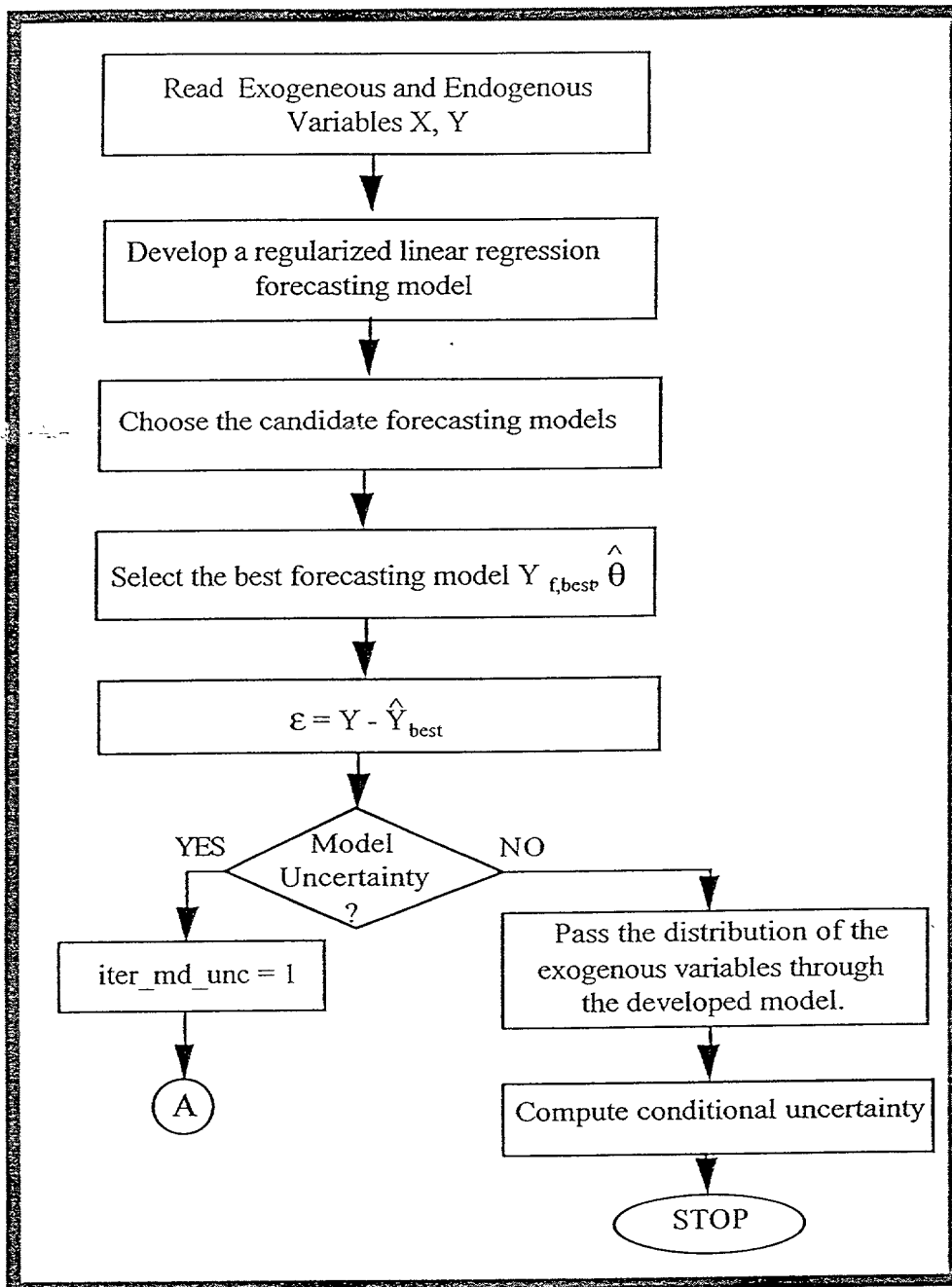


Figure 5.10: Flow Chart of the Uncertainty Assessment Method Using Regularized Linear Regression Point Forecaster.

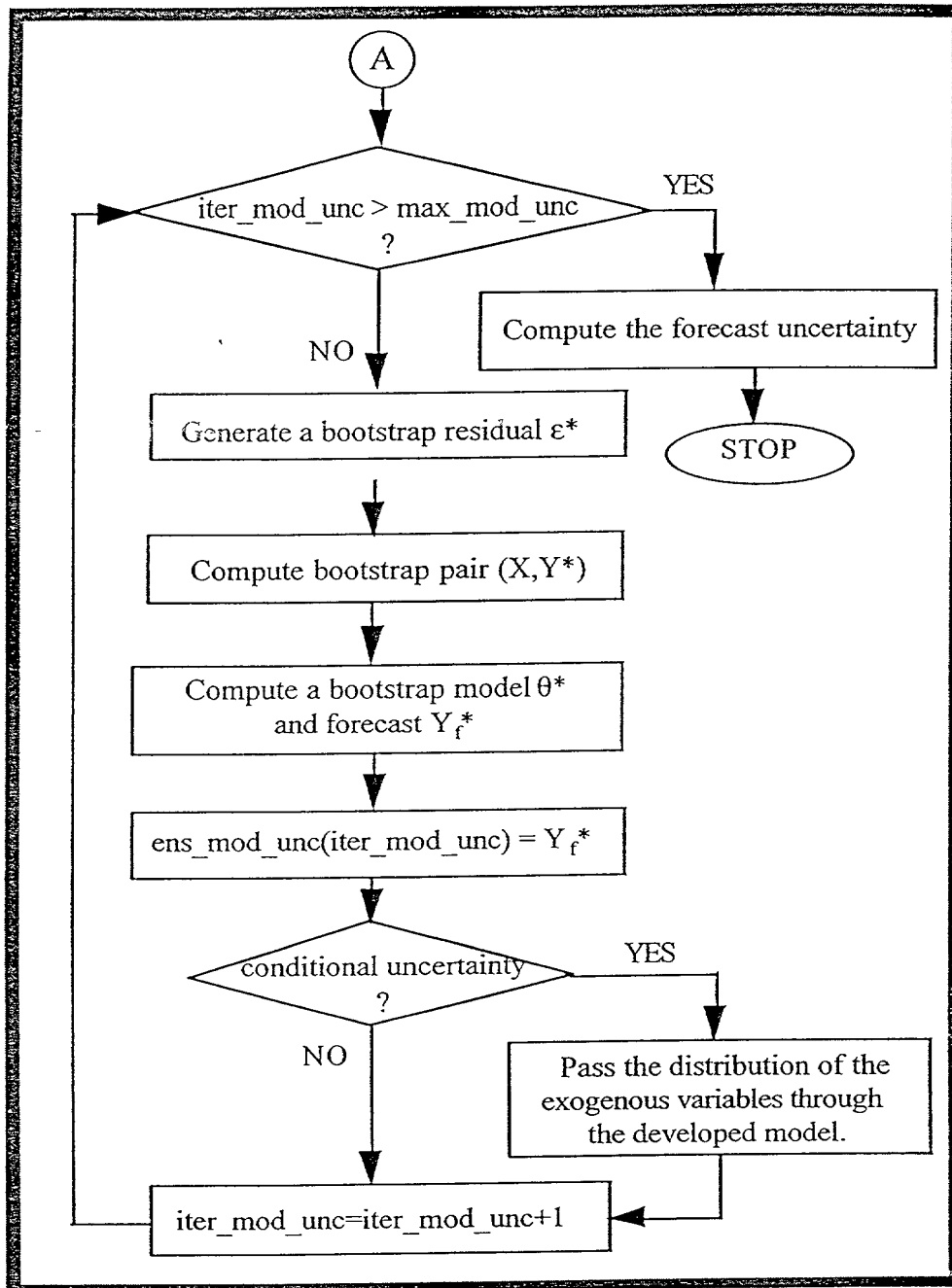


Figure 5.10: Continued.

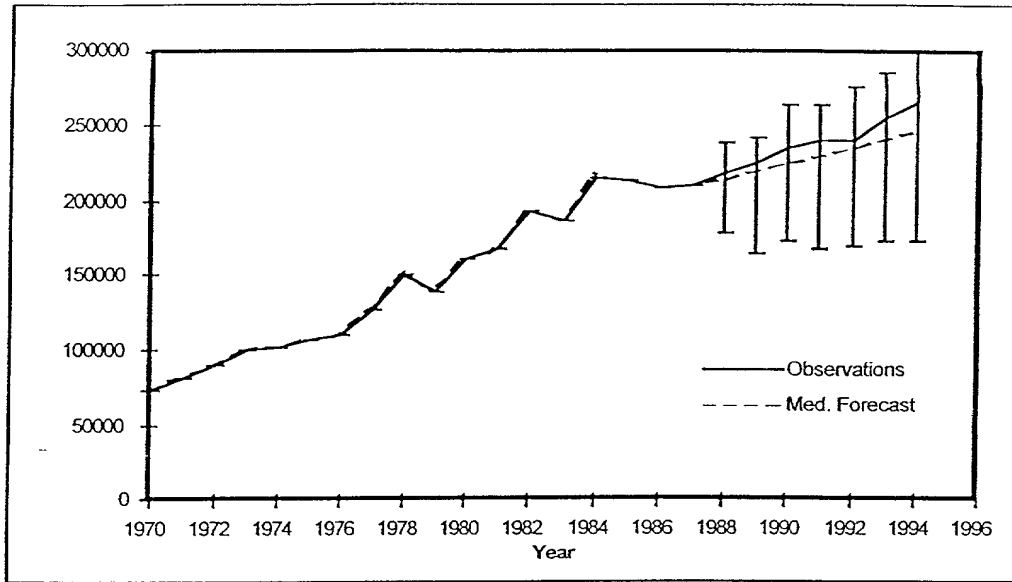


Figure 5.11: Forecast of City Residential Sales and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Linear Regression Point Forecaster.

Figure 5.11 and Table 5.7 show the forecast of the energy sales in the City of Bryan city residential class, forecast errors, and upper and lower 95% confidence bands using a regularized linear regression point forecaster. Average forecast error is 5.1% which is larger than the forecast error using traditional regression models (Figure 5.9 and Table 5.6). The uncertainty level represented as the width of the CI increases as the forecast moves into future. Moreover, the confidence bands around the point forecast are not symmetric. Table 5.7 shows the percentage difference of the lower and upper 95% confidence bands compared to the mean forecast. The point forecast is closer to the upper limit of the confidence band through out forecasting period.

In the year 1988, the upper and lower limits of the 95% confidence band are 12.7% and 16.8% apart from the point forecast respectively. In the year 1988, the 95% CIs on the energy sales represent 30% of the point forecast in the year 1988. This

Table 5.7: Forecast of City Residential Sales , Forecast Error and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Linear Regression Point Forecaster.

Fiscal Year	Sales (MWH)	Forecast and Forecast Error of Energy sales (MWH)					
1970	74030						
1971	82958						
1972	89745						
1973	99952						
1974	103068						
1975	107629						
1976	109980						
1977	127961						
1978	149597						
1979	138895						
1980	159641						
1981	165912						
1982	191520						
1983	185953						
1984	214618						
1985	212105						
1986	207539	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1987	210360						
1988	216769	212363	-2.0	176613	239314	-16.8	12.7
1989	224960	218365	-2.9	164249	241418	-24.8	10.6
1990	235767	223435	-5.2	171731	264872	-23.1	18.5
1991	240156	228693	-4.8	167054	264766	-27.0	15.8
1992	240515	234007	-2.7	168522	276328	-28.0	18.1
1993	255000	239765	-6.0	171775	286024	-28.4	19.3
1994	266036	245575	-7.7	172901	298564	-29.6	21.6
			5.1				

deviation grows through the forecasting period. In the year 1994, the upper and lower limits of the 95% confidence bands are 21.6% and 29.6% apart from the point forecast respectively. In the year 1994, the 95% confidence band on the energy sales represent 51.2% of the point forecast in the same year. Obviously, this shows a large uncertainty on the energy sales compared to the uncertainty bands developed using simple regression models.

V.5.3 Uncertainty Assessment Using Regularized Neural Network Point Forecaster

It was shown in chapter IV, that a number of developed neural network forecasting methods outperformed the linear forecasting models, in both conditional and unconditional forecast of the energy sales in the member utilities of TMPA.

Using a selected stopping criteria, the nonlinear point forecaster searches different model structures and network parameters using a Monte Carlo filtering process and selects a number of candidate model structures and networks. The final energy sales forecast would be the average of forecasts of the top performing models. This method was described in detail in chapter III. In this section, the effect of using such neural network point forecasters on uncertainty assessment is studied.

To use the neural network point forecaster in the uncertainty analysis, a network among the candidate model structures, which has the closest point forecast to the average point forecast is selected. For convenience, this model is called the “best” forecasting model. The prediction of the “best” model, in the historical period is used to generate the original bootstrap residual vector. To assess the model uncertainty, the bootstrap selection with replacement from the residual vector is used to build bootstrap historical input and output pairs. The bootstrap pairs are trained using the same stopping criterion

used in developing the original point forecast. However, the search in the deterministic Monte Carlo filtering process is confined to networks of the same model structure, as the “best” model structure. In developing models for the bootstrap pairs, only a small number of networks are trained in each iteration, since it is required to select only one network which satisfies the acceptable attribute conditions. The forecast of this model generates one realization of the total forecast ensemble. Repeating the bootstrap process for a large number of times will generate an ensemble which can assess the forecasting model parameter uncertainty.

To account for the model parameter uncertainty as well as indicator uncertainty, the future distribution of the exogenous variables will be passed through each of the bootstrap developed models. This will generate an ensemble of forecasts for each bootstrap model. The variability in the final forecast ensemble is an assessment of the model parameter and indicator uncertainty. Figure 5.12 shows a flow chart of the uncertainty assessment method using a regularized neural network point forecaster.

Figure 5.13 and Table 5.8 show the forecast of the energy sales in the City of Bryan city residential class, forecast errors, and upper and lower 95% confidence bands using a regularized neural network point forecaster. Average forecast error is 4.8%, which is larger than the forecast error using traditional regression forecast (Figure 5.9 and Table 5.6) and lower than the regularized linear regression forecast (Figure 5.11 and Table 5.7). The uncertainty level represented as the width of the CI increases as the forecast moves into future. Similar to the regression and regularized linear regression, the confidence bands are not symmetric. Table 5.8 shows the percentage difference of the lower and upper 95% confidence bands compared to the mean forecast. The point forecast is closer to the upper limit of the confidence band through out the forecasting period. In the year 1988, the upper and the lower limits of the 95% confidence bands are

11.7% and 7.0% apart from the point forecast respectively. In the year 1988, the 95% CI on the energy sales represents 18.7% of the point forecast in the year 1988. This deviation grows through the forecasting years. In the year 1994, the upper and lower limits of the 95% confidence bands are 21.6% and 11.0% apart from the point forecast

The following abbreviations have been used in the flow chart:

max_mod_unc	Maximum number of iterations for model and parameter uncertainty computation
max_con_unc	Maximum number of iterations for indicator uncertainty computation
iter_mod_unc	Iteration counter for model and parameter uncertainty computation
iter_con_unc	Iteration counter for indicator uncertainty computation
mod_con_unc_ens	The array which stores the ensemble of forecasts.
ens_mod_unc	The array which stores the ensemble of forecasts, resulting from model uncertainty computation.
ens_con_unc	The array which stores the ensemble of forecasts, resulting from indicator uncertainty computation

Figure 5.12: Flow Chart of the Uncertainty Assessment Method Using Regularized Neural Network Point Forecaster.

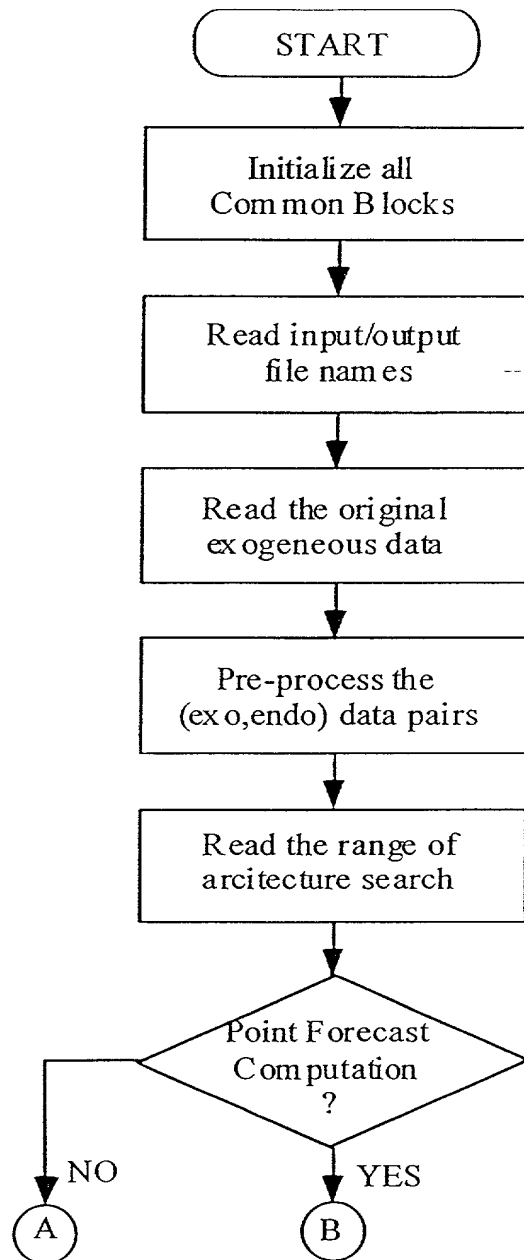


Figure 5.12: Continued.

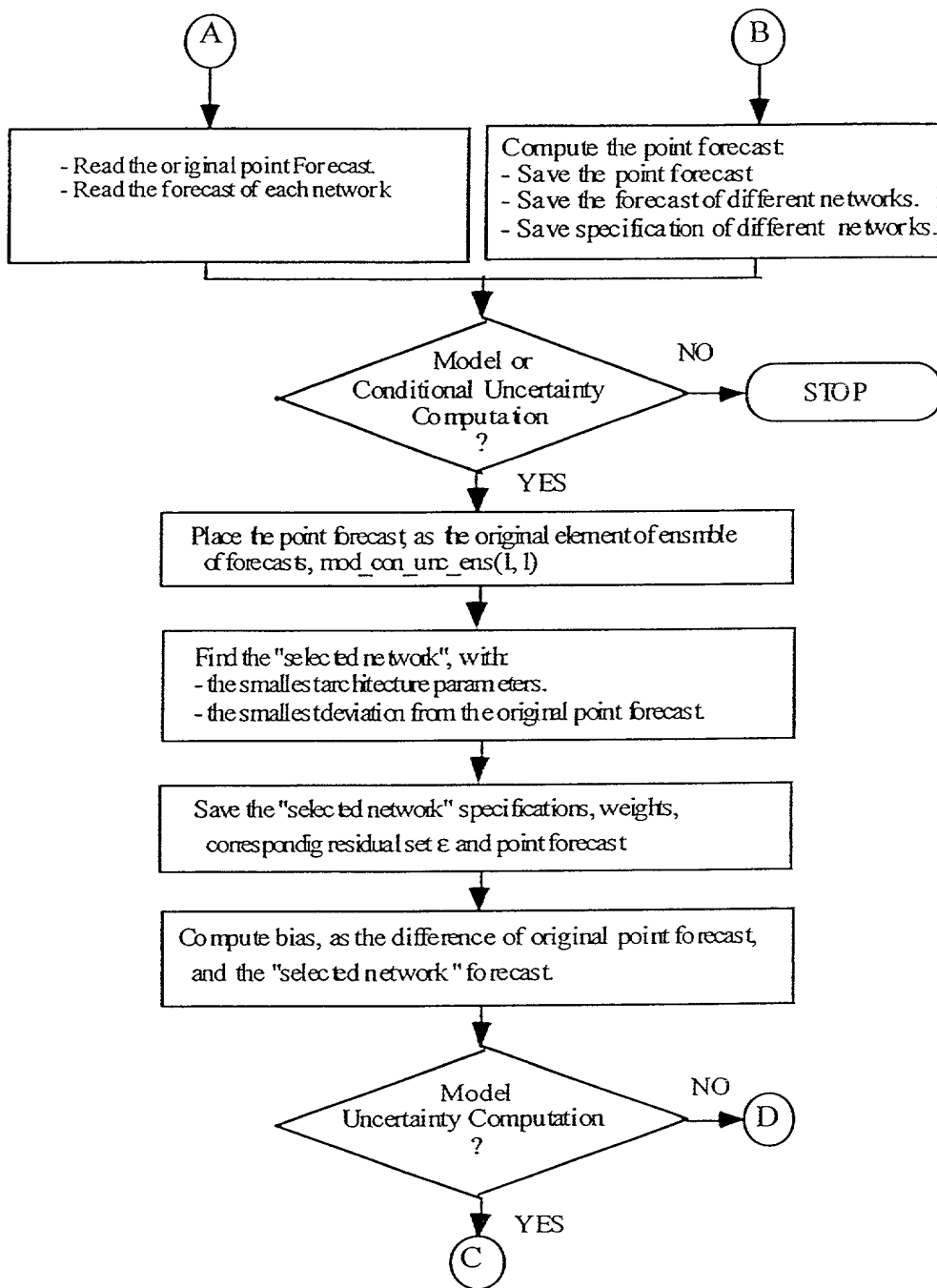


Figure 5.12: Continued.

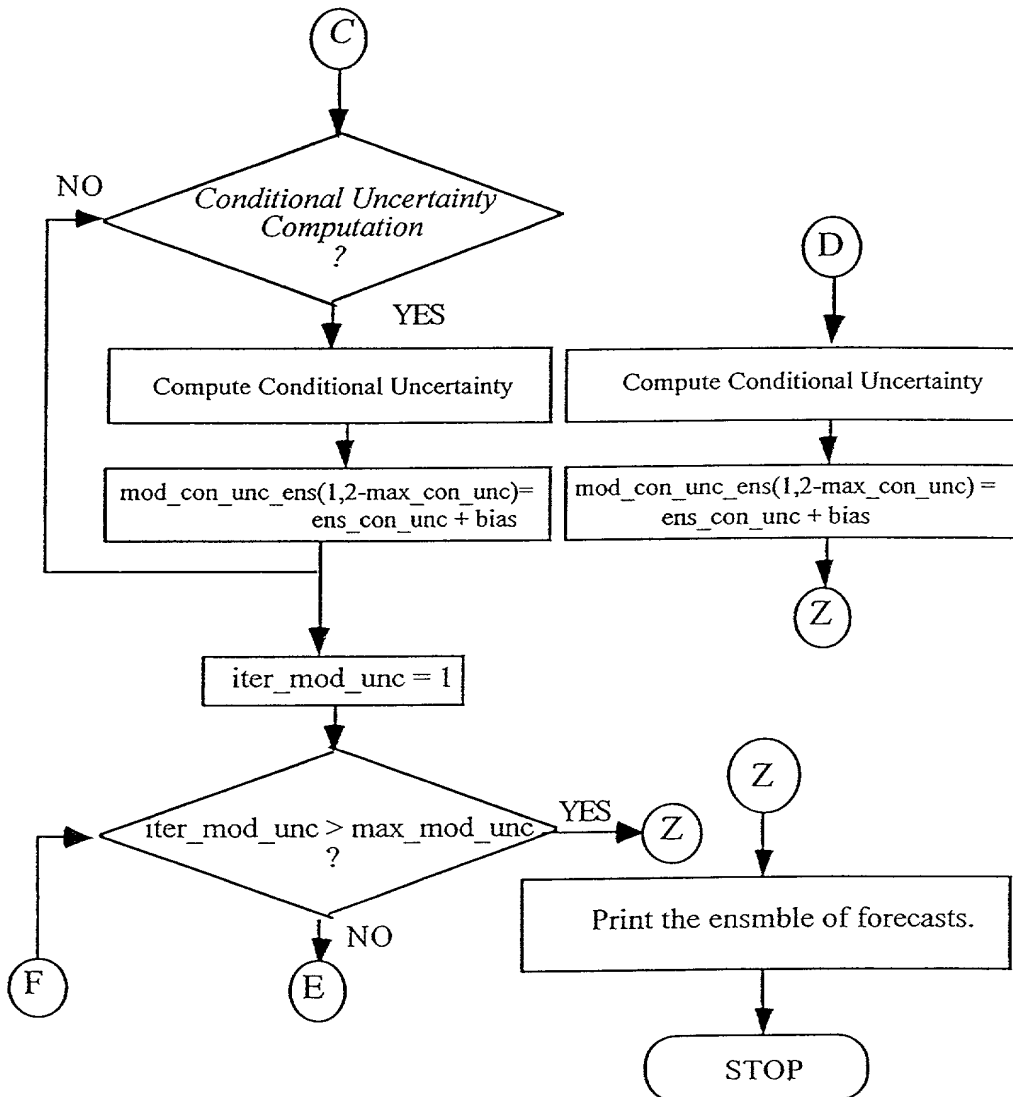


Figure 5.12: Continued.

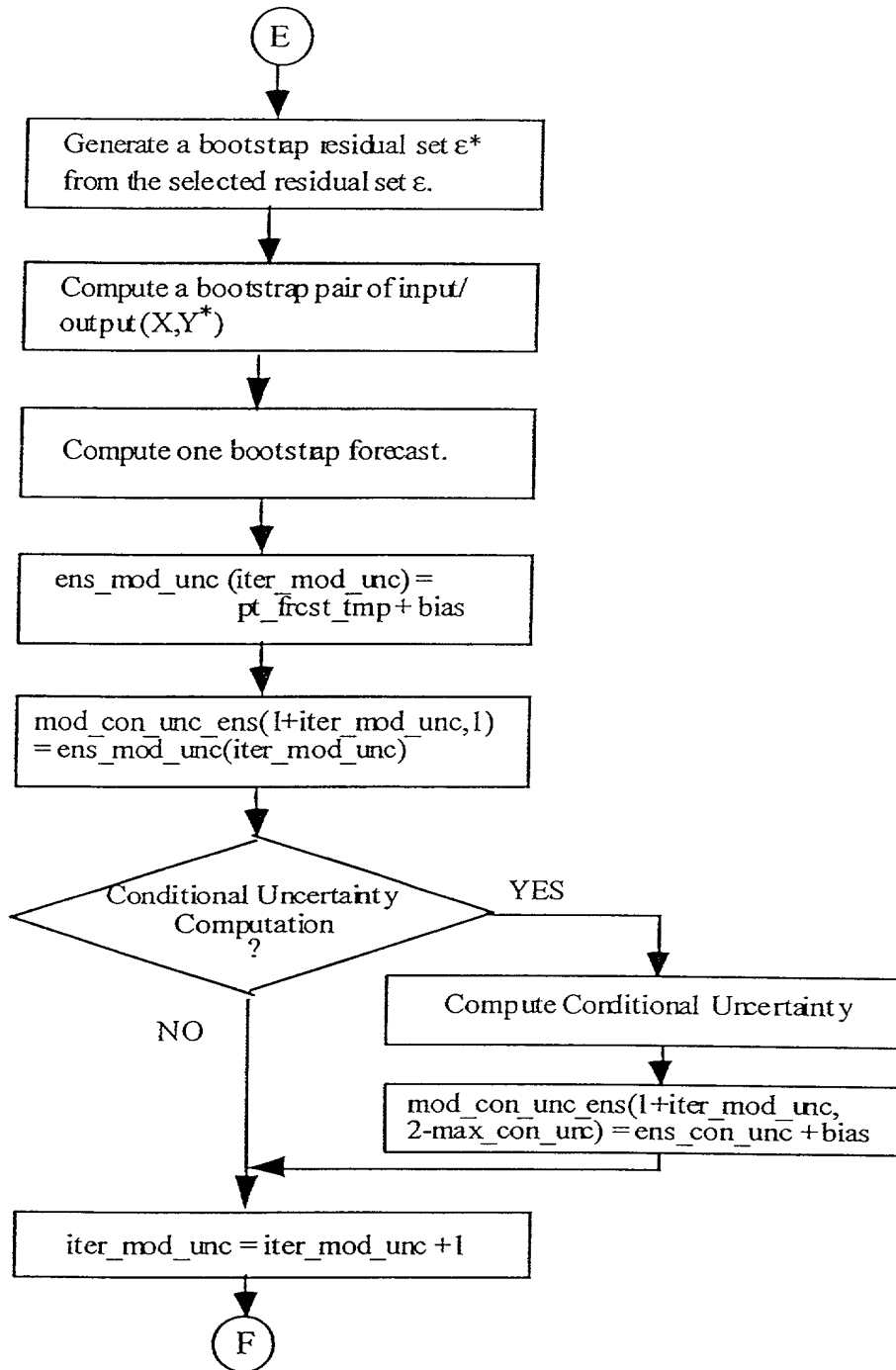


Figure 5.12: Continued.

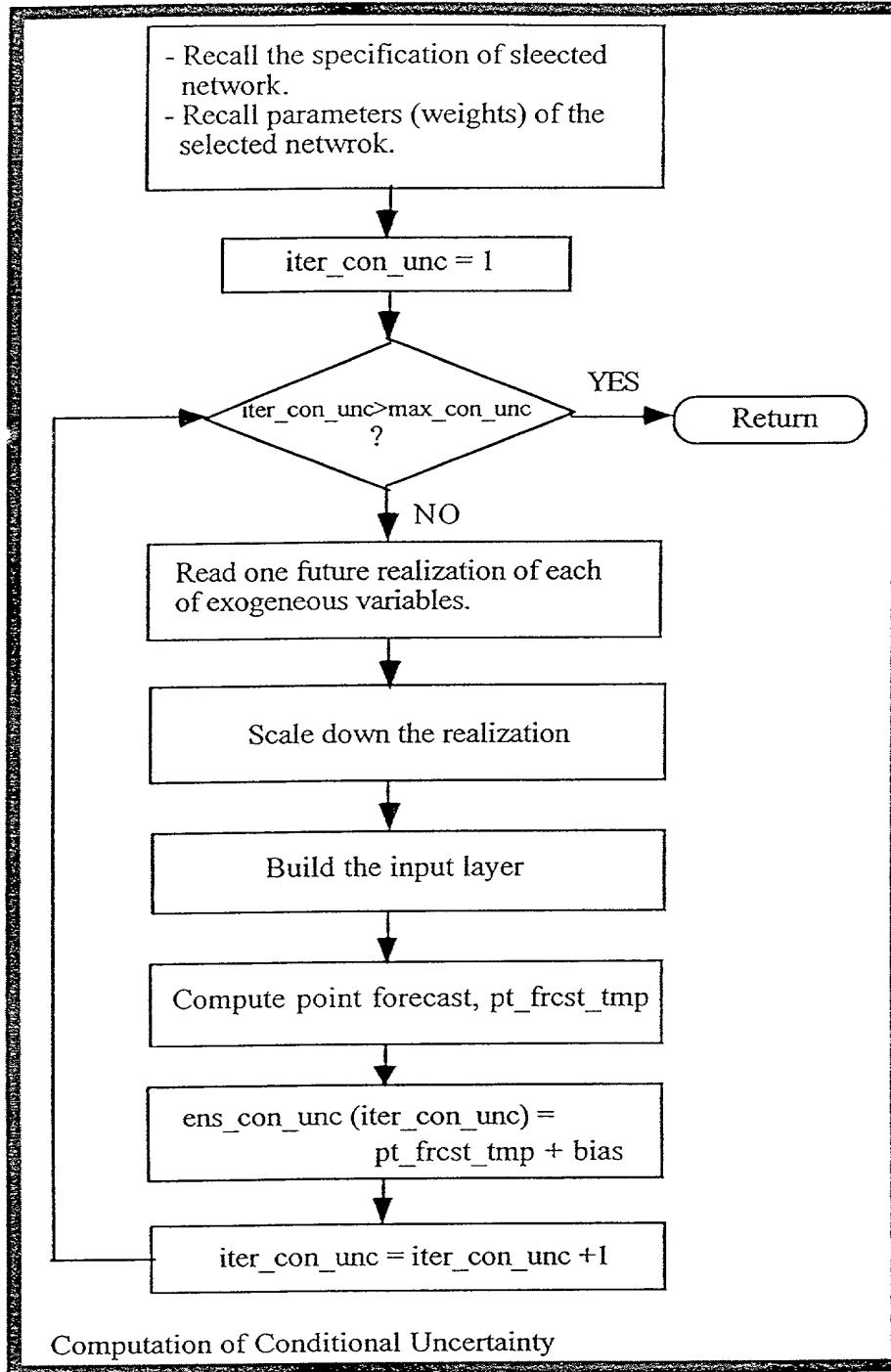


Figure 5.12: Continued.

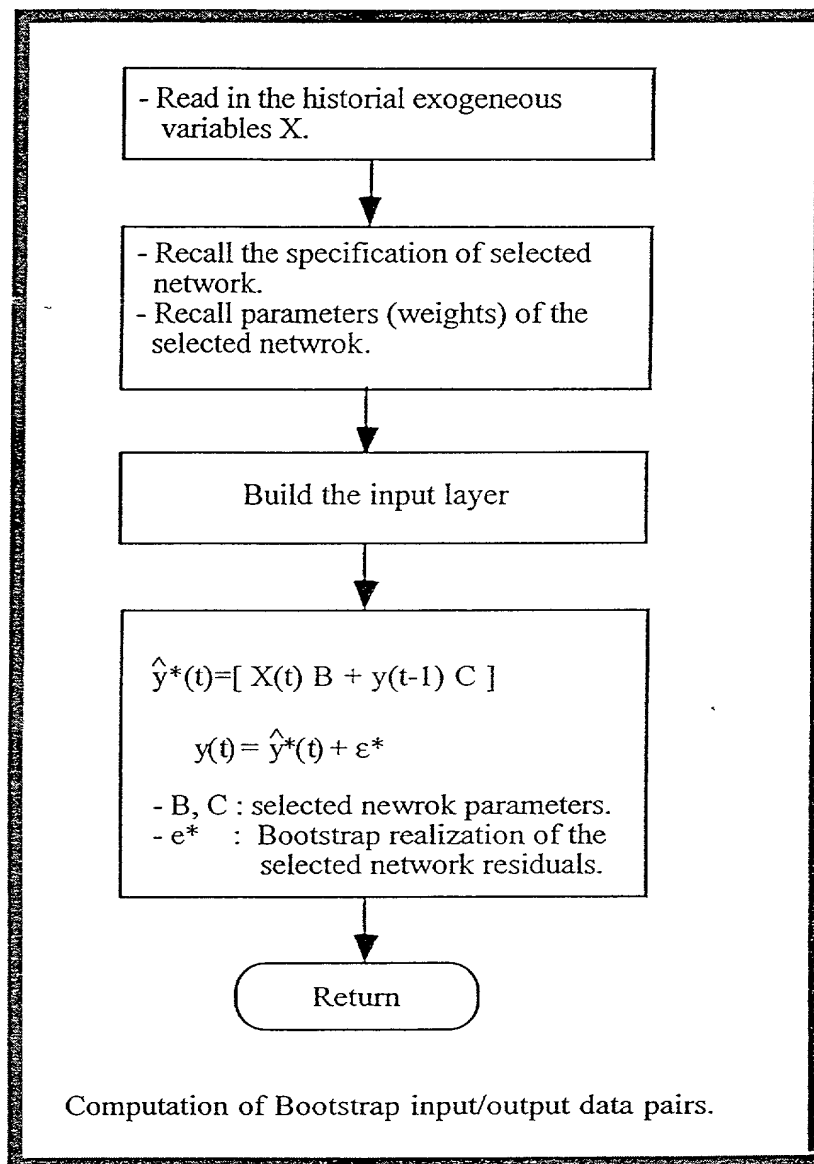


Figure 5.12: Continued.

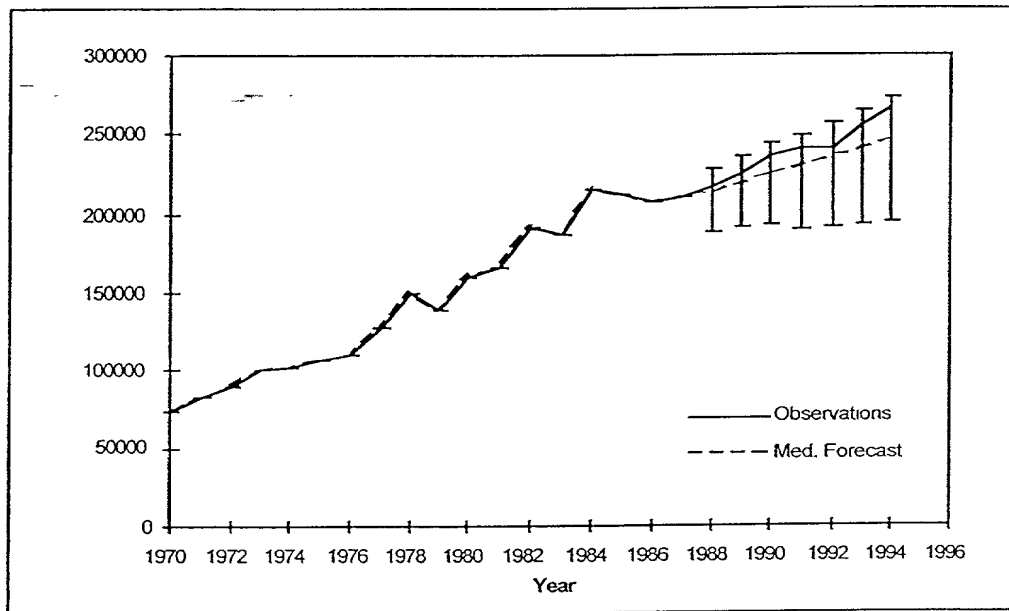


Figure 5.13: Forecast of City Residential Sales and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Neural Network Point Forecaster.

Table 5.8: Forecast of City Residential Sales , Forecast Error and Lower and Upper Limit 95% Confidence Bands in MWH Using Regularized Neural Network Point Forecaster.

Fiscal Year	Sales (MWH)	Forecast and Forecast Error of Energy sales (MWH)					
1970	74030						
1971	82958						
1972	89745						
1973	99952						
1974	103068						
1975	107629						
1976	109980						
1977	127961						
1978	149597						
1979	138895						
1980	159641						
1981	165912						
1982	191520						
1983	185953						
1984	214618						
1985	212105						
1986	207539	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1987	210360						
1988	216769						
1989	224960						
1990	235767						
1991	240156						
1992	240515						
1993	255000						
1994	266036						
			4.8				

respectively. In the year 1994, the 95% confidence band on the energy demand represent 31% of the point forecast in the same year.

V.5.4 Comparison of Confidence Bands Generated By Three Point Forecasters

The intent of the above study was to investigate the effects of point forecast on forecast uncertainty using linear regression and neural network estimators. In chapter IV, it was shown that the regularized neural network models outperformed ordinary regression models in forecasting the energy=sales and peak load demand in the member utilities of TMPA, in 16 out of 18 case studies. These models were developed for the City of Bryan, for the forecasting period from 1984 to 1990. The ordinary regression model for the City of Bryan, city residential class, utilized in the uncertainty analysis was developed for the forecasting period from 1988 to 1994.

The confidence bands generated using ordinary regression models are narrower than those generated by the regularized linear regression and regularized neural network point forecasters. Moreover, it should be noted that the average error of the ordinary regression point forecaster of the energy sales has the minimum error among the three methods. The model with the lowest point forecaster error has the narrowest 95% confidence band. Similarly, the regularized regression model, with the largest point forecast error of 5.1% has the widest 95% confidence band.

The uncertainty level generated by the above methods accounts for both model parameter uncertainty and conditional uncertainty. The model parameter uncertainty represented as CI around the original point forecast is a function of the magnitude of the original residuals. In development of regression forecaster, all the estimation set is used in model development. The basic theme behind this approach is to learn the historical trend as good as possible. The drawback of this approach is that the model learns those

historical events which have no chance of occurring in future. Therefore using regression forecaster, the magnitude of the resulted residuals will comparably be small.

The basic theme behind regularized regression and regularized NN model is to avoid learning historical trends as good as possible. This results in generating more accurate forecasts in the forecasting period. Therefore residuals of the estimation set are larger in magnitude compared to those resulted from ordinary regression forecaster. Larger residual magnitude results in different bootstrap historical sales realizations to have large variance. Large variance among different bootstrap historical observations will result in large variance in the future ensemble of sales realizations. This explains wider confidence bands resulting from the regularized regression and the regularized NN.

Traditionally, it has been a practice in long term load forecasting to associate a 10% band above and below the 10 year forecast of the energy sales, to account for 95% confidence level. Such assumptions were based on accurate model structure and parameter identification and estimation. Recently, some researchers [40] have suggested considering a total band of about 30% of the energy point forecast around the 10 year energy forecast to account for 95% confidence level. However, these researchers have assumed the model parameter uncertainty to follow a normal distribution. A 30% band around the 10 year forecast, is equal to 3% cumulative uncertainty around the point forecast annually. Using the results given in Tables 5.6, 5.7 and 5.8 show that the regression, regularized regression and regularized neural network models introduce on the average 2.5%, 7% and 4.2% cumulative uncertainty bands on the average point forecast respectively. While, the band generated using the regularized regression is extremely large, the bands generated using the regularized neural network in a bootstrapping framework is larger than the reported results.

The selection of the City of Bryan, city residential class for this comparative study does not invalidate the previous results concerning the accuracy of forecasts developed using the ordinary regression and the regularized neural network models. Chapter IV showed that the ordinary regression outperformed the regularized neural network forecasts in only two out of eighteen cast studies. Therefore, it can be strongly argued that although the ordinary regression generates narrower confidence bands, these will be around a point forecast with significant error. On the other hand, although the neural network forecaster generates wider confidence bands, these are around a point forecast which is, in general, more accurate.

The above discussion presents the dilemma between the forecast bias and forecast variance. Decreasing the forecast bias, while keeping the estimator complexity constant, does not necessarily result in a decrease in the forecast variance or in other words narrower confidence bands.

All the above uncertainty assessment methods have utilized the bootstrap techniques to develop confidence bands around the point forecast of the energy sales. The above analysis shows that the choice of the point forecaster is an important factor in accuracy of the generated confidence bands. Based on the above analysis and the evidence that the regularized neural networks result in more accurate point forecasts, it can be concluded that regularized neural networks are viable estimators for use in uncertainty assessment analysis.

V.6 Chapter Summary

This chapter introduced and reviewed the statistical resampling plans, and in particular the bootstrapping methods.

The application of the bootstrapping methods in assessing the accuracy of a particular estimate, based only on the data at hand was presented. The principle of bootstrap method to assess the uncertainty in estimation of a particular statistics was explained through its application to the sample mean. The result of application of bootstrap was compared favorably to other estimates of the uncertainty.

Through extensive computations, the bootstrap tries to establish inference in a nonparametric context, therefore relieving the forecaster from unrealistic parametric assumptions. The computation of confidence bands require adopting nonparametric confidence band estimation techniques. Several published nonparametric confidence band computation methods were discussed, and their advantage and drawbacks were pointed. The reasons behind adopting the “Minimum Length Method” were explained.

To generate the ensemble of forecasts, the bootstrap depends heavily on the point forecaster. The uncertainty bands generated using three different point forecasters, namely: regression models, regularized linear regression models, and regularized neural network models were presented and compared. Implementation details and flow chart of each of the above uncertainty assessments methods were explained in detail.

While the actual demand of the energy fell within the 95% confidence bands forecasted by all the above three methods, the results showed that the width of the confidence bands depends on the type of the point forecaster selected for the point forecast estimates.

CHAPTER VI

UNCERTAINTY ASSESSMENT OF FORECASTS FOR CITY OF BRYAN SYSTEM

VI.1 Introduction

The City of Bryan system is divided into a city division and a rural division. The city division sales include a residential class, small, medium and a large commercial class, a school class and an interdepartmental class. The three commercial classes have been combined into a single city commercial class for the purpose of this study because of the sparsity of the available data. The contribution of the school class and of the interdepartmental class to the total system sales is less than 5%, therefore no models were developed for these two classes. The rural division of the City of Bryan includes a residential class and a commercial class. Forecasting uncertainty was modeled and studied for energy sales in each of the customer classes mentioned. Also, forecasting uncertainty was modeled and studied for the total system peak load demand.

Uncertainty assessments of sales forecasts in different classes of customers account for both conditional and parameter uncertainties. The term conditional uncertainty refers to uncertainty in future sales forecasts due to uncertainties in future values of the exogenous (input) variables. Model parameter uncertainty refers to uncertainty in future energy sales forecasts due to uncertainty in the estimated forecasting model parameters.

This chapter is organized as follows. Section VI.2 briefly describes the uncertainty methods developed to assess the forecast uncertainty in the City of Bryan. Section VI.3 presents the computed confidence bands on forecasts of energy sales for different class of customers, the total system energy sales and peak load demand in the City of Bryan. Section VI.4 analyzes the results presented in section VI.3. Section VI.5 summarizes the chapter.

VI.2 Uncertainty Assessment Method

In chapter III several nonlinear forecasting methods were developed. Chapter IV showed that these methods generally outperformed the available linear regression models in forecasting energy sales and peak load demand in the member utilities of TMPA. In chapter V it was shown that bootstrapping techniques result in accurate estimates of the confidence bands around forecasts of energy sales.

The result of the Monte Carlo filtering process is a set of model structures which satisfy the acceptable accuracy requirements. The final forecast of energy sales is an average of the forecasts obtained from the methods in the top performing model sets. The forecasts obtained from each of the top performing models generate a residual vector in the historical period. However, the bootstrap method requires a single residual vector. Therefore a single residual vector must be selected from the residual vectors of the top performing forecasting models. This is done by choosing the residual vector of the “best” forecasting model. The “best” forecasting model is the model which has its forecast closest to the average forecast of the top performing forecasting models. The forecast of the “best” model in the historical period is used to generate the original bootstrap residual vector. Using selection with replacement from this residual vector, bootstrap historical input and output pairs are generated. The bootstrap pairs are used to develop

bootstrap models using the same stopping criterion used in developing the original point forecast. However, the search in the deterministic Monte Carlo filtering process is confined to networks of the same model structure as the “best” model structure. In developing models using the bootstrap pairs only a small number of networks are trained in each iteration since it is required to select only one network which satisfies the acceptable accuracy conditions. Using the mean forecast values of the exogenous variables, the bootstrap model generates a single forecast of future energy sales. Repeating the bootstrap process a large number of times generates an ensemble of energy sales forecasts. The variability in the generated forecast ensemble can be used to assess the uncertainty in future energy sales forecasts due to uncertainty in the computed model parameters.

To account for exogenous variables and model parameter uncertainty, the forecast distributions of the exogenous variables are passed through each of the bootstrap developed models. This generates an ensemble of forecasts for each bootstrap model. The variability in the final forecast ensemble is then attributed to model parameter and exogenous variables uncertainties.

Confidence bands provide measures of uncertainty in the sales forecasts. The empirical distributions of the “best” model residuals do not follow any particular distribution form. Therefore, nonparametric methods should be used to compute the confidence bands of the future energy sales ensemble. Nonparametric methods require a minimum number of ensemble realizations for a particular confidence level. Obviously, as the required confidence level increases, the required number of ensemble realizations increases. It is common to compute 95% confidence bands for long-term load forecasts.

Reliably assessing the model uncertainty for a particular confidence level requires convergence of the nonparametric confidence interval computation method. Experience

has shown that for a confidence level of 95% the required number of bootstrap realizations is at least 500. The number of bootstrap realizations required, depends on the type of estimator, the variability of the model parameters and the distribution of the future exogenous variable forecasts.

VI.3 Uncertainty Assessment of Energy Sales Forecasts in Customer Classes of City of Bryan

Historical observations of annual energy sales in each of the different classes of customers and of the corresponding exogenous variables were available for the 25 year period from 1970 to 1994.

The forecasting (backcasting) period was chosen to be the seven most recent years of the observations. The remaining 18 years of historical data constitute the estimation set. The estimation set is further divided into a training set and a validation set. The training set includes 13 years of historical observation from 1970 to 1982. The validation set includes the remaining 5 years of the observation from 1983 to 1987.

VI.3.1 City Commercial Class Sales Forecast Uncertainty Model

Three exogenous variables have been used in forecasting the sales in the city commercial class (Section IV3.3). These are time, the number of commercial customers and the price of electricity. The conditional uncertainty in the sales forecast is due to uncertainty in the exogenous variables as well as uncertainty in the parameters of the forecasting model.

The future time variable does not include any uncertainty. The general practice in forecasting future number of customers is to specify High, Medium and Low

projections. The City of Bryan projections for High, Medium and Low customers growth rates were 4%, 2.3% and 1.5% average annual growth based on 1987 commercial customers, respectively. This projection sets the medium forecast to be the most probable future scenario. It was decided to model these projections as a normal distribution with High, Medium and Low projections representing the 97.5%, 50% and 2.5% cumulative probability levels, respectively. That is the probability that the customers growth rate is

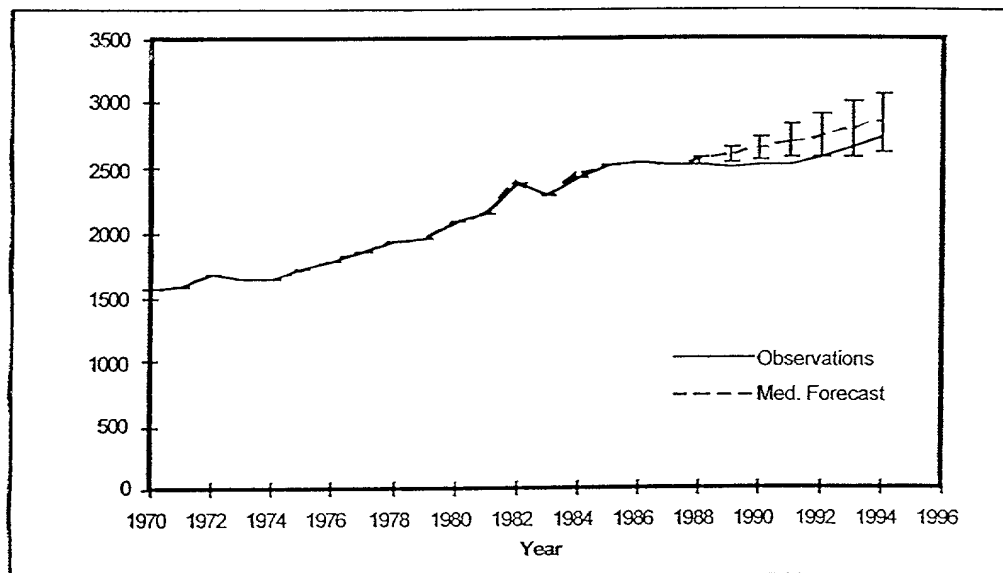


Figure 6.1: Forecast of the City Commercial Customers and 95% Confidence Bands

higher than the High forecast or lower than the Low forecast is 2.5% and the customers growth rate is equally likely to be higher or lower than the Medium forecast. Figure 6.1 shows the historical number of customers time series as well as the forecast uncertainty level as defined above. Clearly, the actual number of customers in the years 1988 to 1992 is outside of the 95% forecast uncertainty bands on the projected number of customers. That is, the forecast of number of customers by the City of Bryan were rather seriously in error.

Similar to the projection of the commercial customers, the general practice in forecasting future values of price of electricity is to specify High, Medium, and Low projections of future electricity price in cents per KWh. City of Bryan forecasts for above projections based on the 1987 price were 4.45%, 2.9% and 1.85%, respectively. This projection sets the medium forecast to be the most probable future scenario. It was decided to model these projections as a normal distribution with High, Medium, and

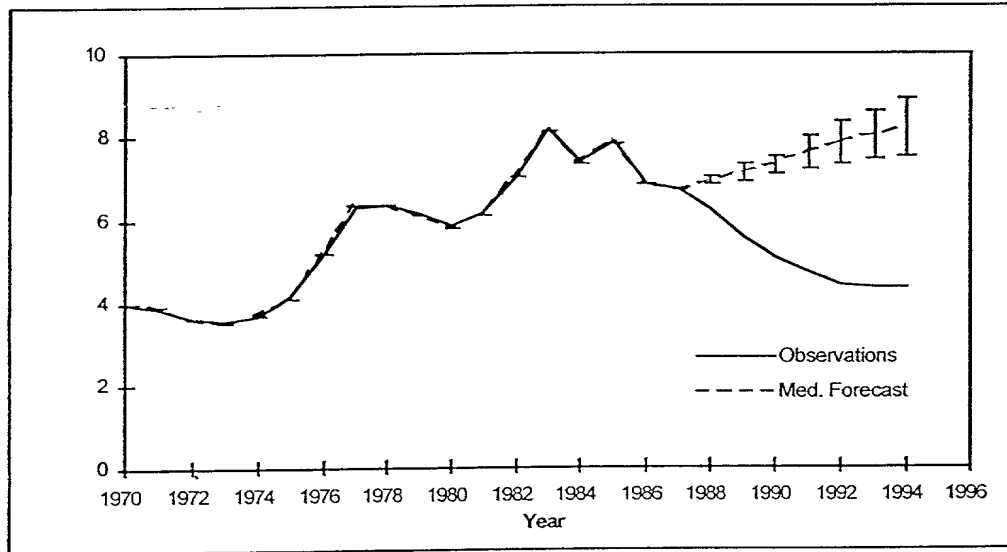


Figure 6.2: Forecast of City Commercial Price (cents per KWh) and 95% Confidence Bands.

Low projections representing 97.5%, 50%, and 2.5% cumulative probability levels, respectively. That is the probability that the price increase is higher than the High forecast or lower than the Low forecast is 2.5% and the price growth rate is equally likely to be higher or lower than the Medium forecast. Figure 6.2 shows the historical price of electricity in cents per KWh time series as well as the forecast uncertainty levels as defined above. Clearly, the actual price of electricity throughout the forecasting period is outside of the 95% confidence bands of the projected price in 1987.

Figure 6.3 and Table 6.1 show the conditional forecast of energy sales in the City of Bryan city commercial class, the forecast errors, and the upper and lower 95% confidence bands on the forecasts. The uncertainty in the energy sales forecast is due to exogenous variables as well as model parameter uncertainties. The average forecast error is 4.4% for the 1988-1994 period. The maximum forecast error was 6.3% in year 1994.

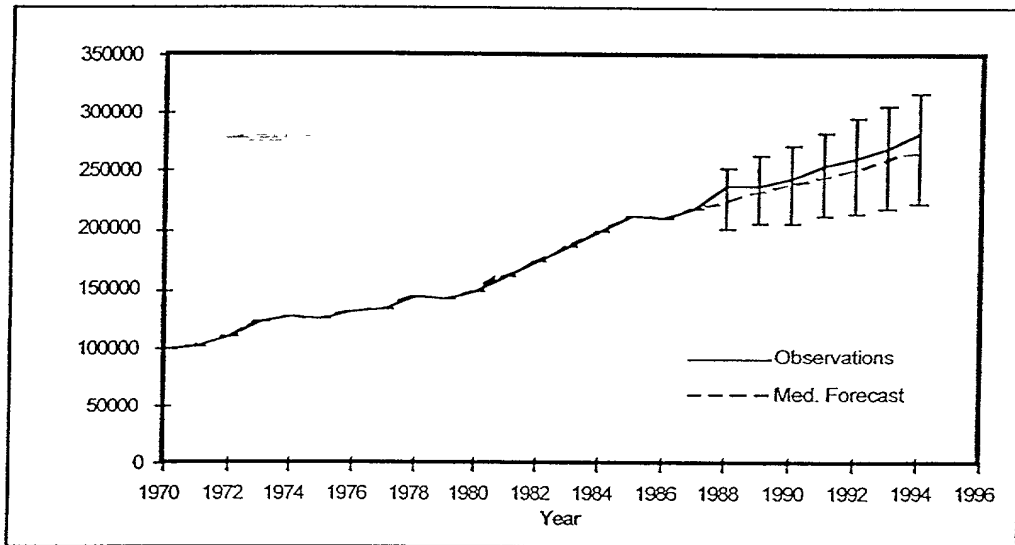


Figure 6.3: Forecast of City Commercial Sales and Lower and Upper 95% Confidence Bands in MWH.

The uncertainty in future energy sales as represented by the width of the 95% confidence band increases as the forecast moves into the future. This agrees with the intuition. Table 6.1 shows the percentage difference of the lower and upper 95% confidence band limits with respect to the mean forecast. The confidence bands are not symmetric around the mean forecast. The mean forecast is closer to the upper limit of the confidence band throughout the forecasting period. In the year 1988, the upper and the lower limits of the 95% confidence band are 12.5% and 10.6% above and below the mean forecast, respectively. In the year 1988, the 95% confidence band on the energy

920

sales amounts to 23.1% of the average forecast. This percentage grows through the forecasting period. In the year 1994, the upper and lower limits of the 95% confidence band are 19.4% and 16.3% above and below the mean forecast, respectively, and the 95% forecast confidence band on the energy demand amounts to 35.7% of the sales forecast in the same year.

Table 6.1: Forecast of City Commercial Class Sales, Forecast Error and 95% Lower and Upper Limit Confidence Bands in MWH.

Fiscal Year	Sales (MWH)	Forecast and Forecast Error of Energy sales (MWH)					
1970	99120						
1971	100869						
1972	109714						
1973	122325						
1974	126752						
1975	125547						
1976	131346						
1977	133379						
1978	144955						
1979	143163						
1980	148894						
1981	162018						
1982	173728						
1983	187200						
1984	199576						
1985	213382						
1986	210589	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1987	219916						
1988	238569	225447	-5.5	201642	253696	10.6	12.5
1989	239940	232438	-3.1	205074	263676	11.8	13.4
1990	246333	238963	-3.0	207549	272977	13.1	14.2
1991	255111	246135	-3.5	212214	284647	13.8	15.6
1992	262217	252807	-3.6	215641	295256	14.7	16.8
1993	270180	259594	-3.9	219917	307030	15.3	18.3
1994	284230	266260	-6.3	222921	317878	16.3	19.4
			4.4				

Above discussion shows that the forecast of energy sales in the city commercial class includes a large level of uncertainty.

VI.3.2 Rural Residential Class Sales Forecast Uncertainty Model

Four exogenous variables have been used in forecasting the sales in the rural residential class (Section IV3.3). These are time, the number of residential customers, CDDs and the price of electricity. The conditional uncertainty in the sales forecast in the rural residential class is due to uncertainty in the exogenous variables as well as uncertainty in the parameters of the forecasting model.

The future time variable does not include any uncertainty. The general practice in forecasting future number of customers is to specify High, Medium and Low projections. The City of Bryan projections for High, Medium and Low customers growth rates were 3%, 1.5% and 0.75% average annual growth based on 1987 rural residential customers, respectively. This projection sets the medium forecast to be the most probable future scenario. It was decided to model these projections as a normal distribution with High, Medium and Low projections representing the 97.5%, 50% and 2.5% cumulative probability levels, respectively. That is the probability that the customers growth rate is higher than the High forecast or lower than the Low forecast is 2.5% and the customers growth rate is equally likely to be higher or lower than the Medium forecast. Figure 6.4 shows the historical number of customers time series as well as the forecast uncertainty level as defined above. Clearly, the City of Bryan forecasted the number of customers in the rural residential class with good accuracy.

Due to unavailability of any forecast of the CDDs in the service area, it was decided to forecast the CDDs in the service area as a time series, with its forecast value

being the average of the 18 years of the historical CDDs. The graph of the historical CDDs and the corresponding forecast was given in Figure 5.6 and Table 5.4.

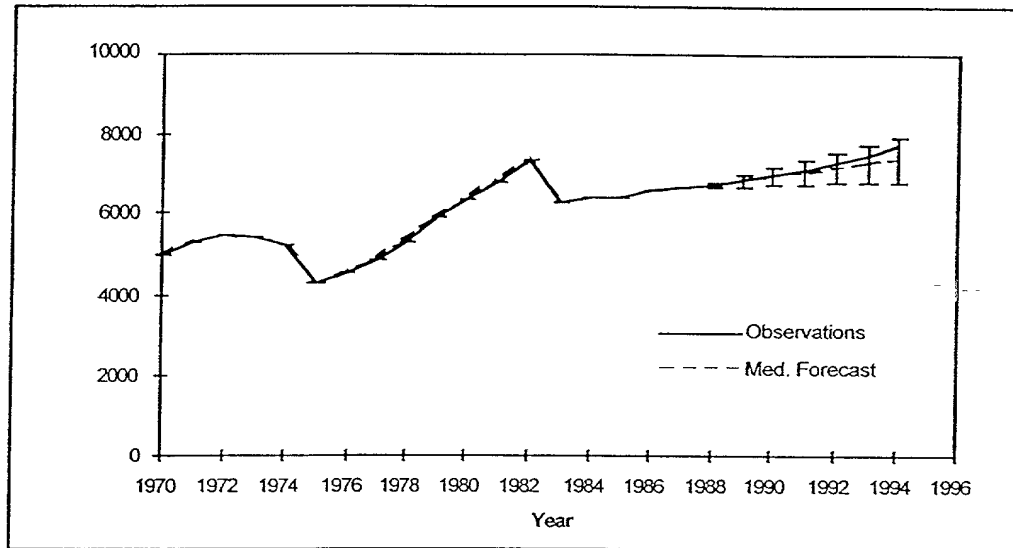


Figure 6.4: Forecast of the Rural Residential Customers and 95% Confidence Bands.

Similar to the projection of the rural residential customers, the general practice in forecasting future values of price of electricity is to specify High, Medium, and Low projections of future electricity price in cents per KWh. City of Bryan forecasts for above projections based on the 1987 price were 4.45%, 2.9% and 1.85%, respectively. This projection sets the medium forecast to be the most probable future scenario. It was decided to model these projections as a normal distribution with High, Medium, and Low projections representing 97.5%, 50%, and 2.5% cumulative probability levels, respectively. That is the probability that the price increase is higher than the High forecast or lower than the Low forecast is 2.5% and the price growth rate is equally likely to be higher or lower than the Medium forecast. Figure 6.5 shows the historical

924

price of electricity in cents per KWh time series as well as the forecast uncertainty levels as defined above. Clearly, the actual price of electricity in the rural residential class throughout the forecasting period is outside of the 95% confidence bands of the projected price in 1987.

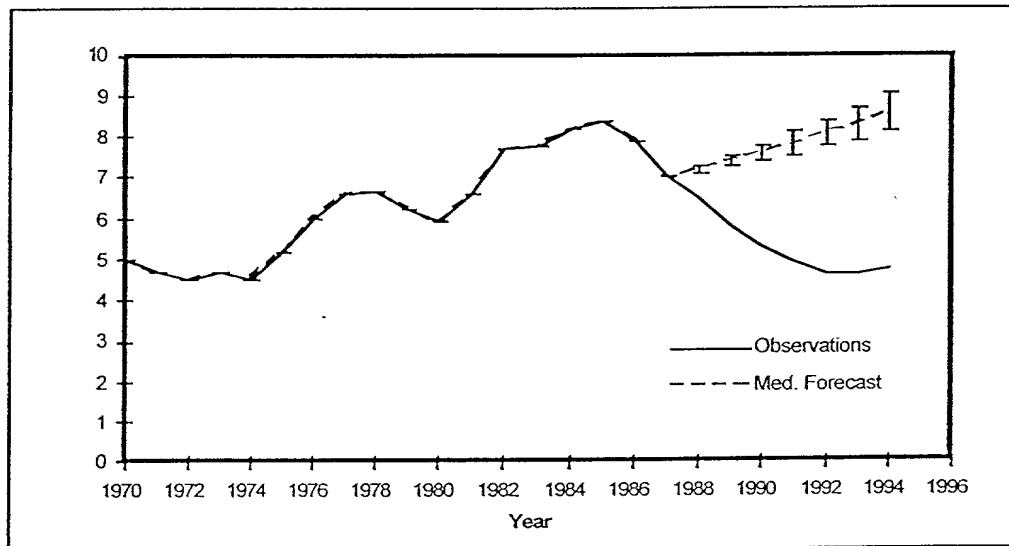


Figure 6.5: Forecast of Rural Residential Price (cents per KWh) and 95% Confidence Bands.

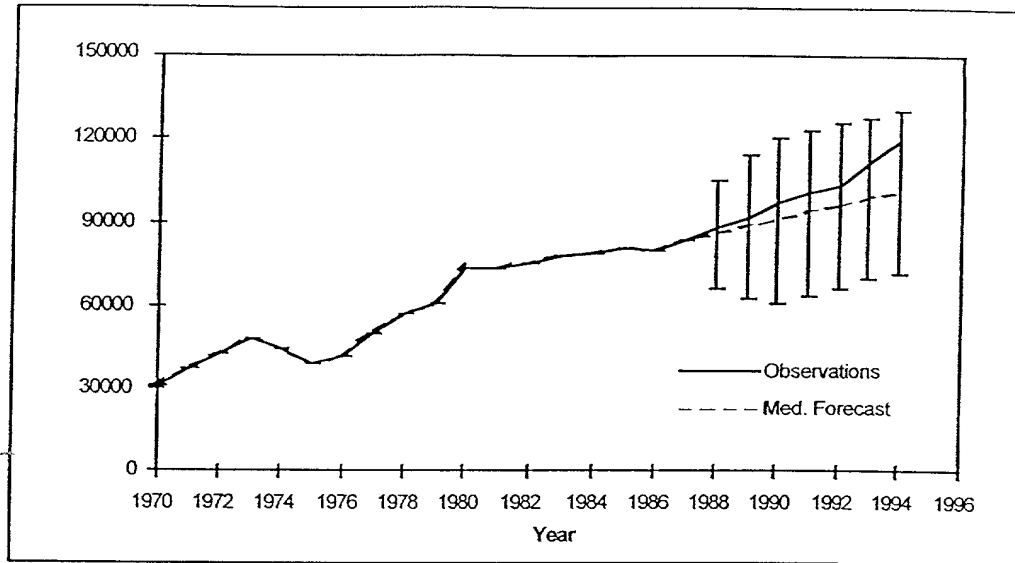


Figure 6.6: Forecast of Rural Residential Sales and Lower and Upper 95% Confidence Bands in MWH.

Figure 6.6 and Table 6.2 show the conditional forecast of energy sales in the City of Bryan rural residential class, the forecast errors, and the upper and lower 95% confidence bands on the forecasts. The uncertainty in the energy sales forecast is due to exogenous variables as well as model parameter uncertainties. The average forecast error is 10.1% for the 1988-1994 period. The maximum forecast error was 15.2% in the year 1994. The uncertainty in future energy sales as represented by the width of the 95% confidence band does not increase as the forecast moves into the future. Table 6.2 shows the percentage difference of the lower and upper 95% confidence band limits with respect to the mean forecast. In the year 1988, the upper and the lower limits of the 95% confidence band are 22.7% above and below the mean forecast, respectively. In the year 1988, the 95% confidence band on the energy sales amounts to 45.4% of the average forecast. In the year 1994, the 95% forecast confidence band on the energy demand amounts to 58.8% of the sales forecast in the same year.

Table 6.2: Forecast of Rural Residential Class Sales, Forecast Error and 95% Lower and Upper Limit Confidence Bands in MWH.

Fiscal Year	Sales (MWH)	Forecast and Forecast Error of Energy sales (MWH)					
1970	31175						
1971	37291						
1972	43000						
1973	48480						
1974	44807						
1975	39050						
1976	41278						
1977	49582						
1978	57123						
1979	60932						
1980	73200						
1981	73443						
1982	75300						
1983	77650						
1984	78993						
1985	80093						
1986	79157	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1987	83672						
1988	87814	85569	-2.6	66136	105002	22.7	22.7
1989	91063	88394	-2.9	63145	113643	28.6	28.6
1990	96775	90810	-6.2	61349	120271	32.4	32.4
1991	100685	93719	-6.9	64117	123321	31.6	31.6
1992	103642	96168	-7.2	66306	126030	31.1	31.1
1993	110546	98573	-10.8	69414	127732	29.6	29.6
1994	118690	100646	-15.2	71106	130186	29.4	29.4
		10.1					

The above discussion shows that the uncertainty in the energy sales forecast in rural residential class is very high. This high level is due to uncertainty in the developed forecasting model as well as uncertainty in the future price of electricity.

VI.3.3 Rural Commercial Class Sales Forecast Uncertainty Model

Three exogenous variables have been used in forecasting the sales in the rural commercial class (Section IV3.3). These are time, the number of commercial customers and the price of electricity. The conditional uncertainty in the sales forecast is due to uncertainty in the exogenous variables as well as uncertainty in the parameters of the forecasting model.

The future time variable does not include any uncertainty. The general practice in forecasting future number of customers is to specify High, Medium and Low projections. The City of Bryan projections for High, Medium and Low customers growth rates were 3%, 2.75% and 2% average annual growth based on 1987 commercial customers, respectively. This projection sets the medium forecast to be the most probable future scenario. It was decided to model these projections as a normal distribution with High, Medium and Low projections representing the 97.5%, 50% and 2.5% cumulative probability levels, respectively. That is the probability that the customers growth rate is higher than the High forecast or lower than the Low forecast is 2.5% and the customers growth rate is equally likely to be higher or lower than the Medium forecast. Figure 6.7 shows the historical number of customers time series as well as the forecast uncertainty level as defined above. Clearly, the actual number of customers in the rural commercial class in the years 1988 to 1992 is outside of the 95% forecast uncertainty bands on the projected number of customers.

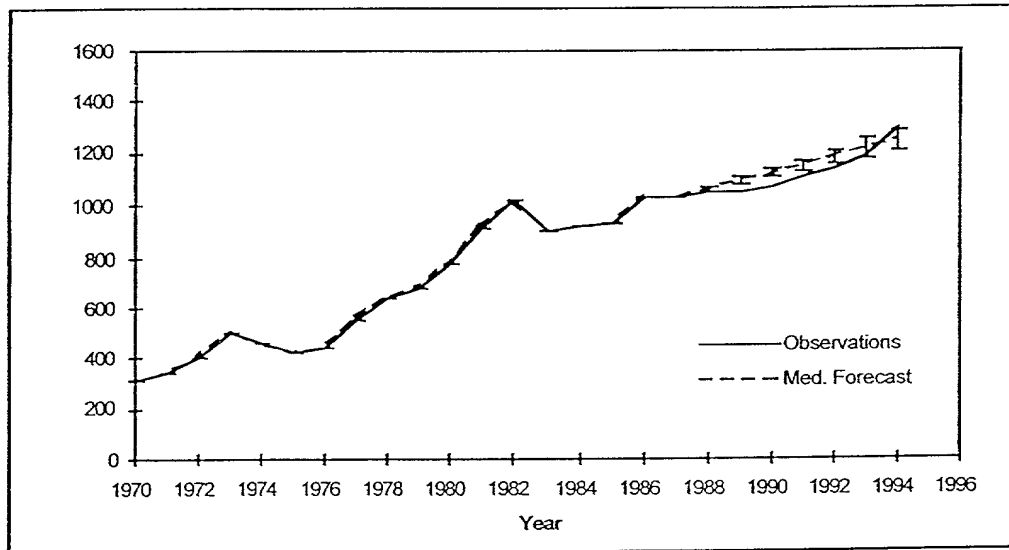


Figure 6.7: Forecast of the Rural Commercial Customers and 95% Confidence Bands.

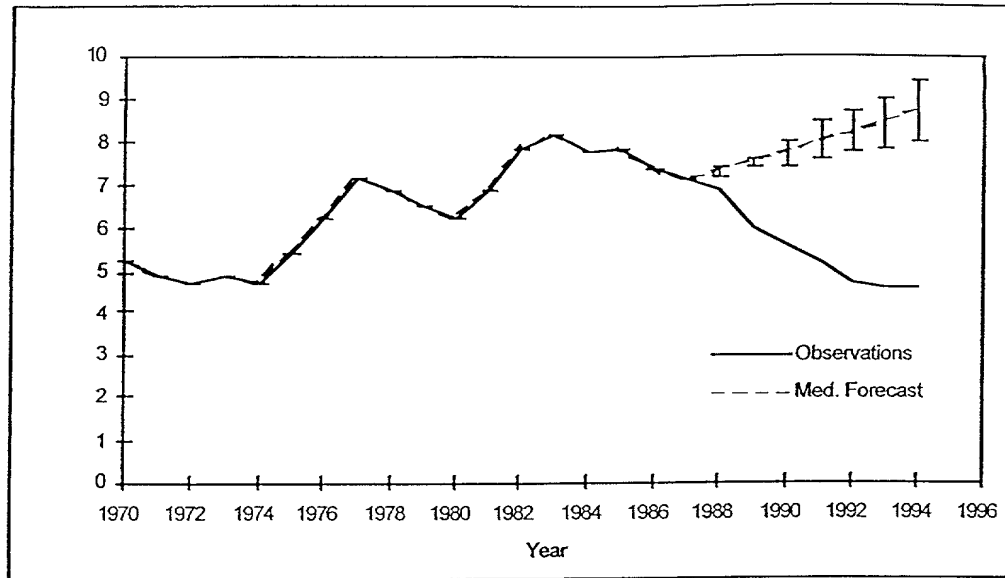


Figure 6.8: Forecast of Rural Commercial Price (cents per KWh) and 95% Confidence Bands.

Similar to the projection of the commercial customers, the general practice in forecasting future values of price of electricity is to specify High, Medium, and Low projections of future electricity price in cents per KWh. City of Bryan forecasts for above projections based on the 1987 price were 4.45%, 2.9% and 1.85%, respectively. This projection sets the medium forecast to be the most probable future scenario. It was decided to model these projections as a normal distribution with High, Medium, and Low projections representing 97.5%, 50%, and 2.5% cumulative probability levels, respectively. That is the probability that the price increase is higher than the High forecast or lower than the Low forecast is 2.5% and the price growth rate is equally likely to be higher or lower than the Medium forecast. Figure 6.8 shows the historical price of electricity in cents per KWh time series as well as the forecast uncertainty levels as defined above. Clearly, the actual price of electricity throughout the forecasting period is outside of the 95% confidence bands of the projected price in 1987.

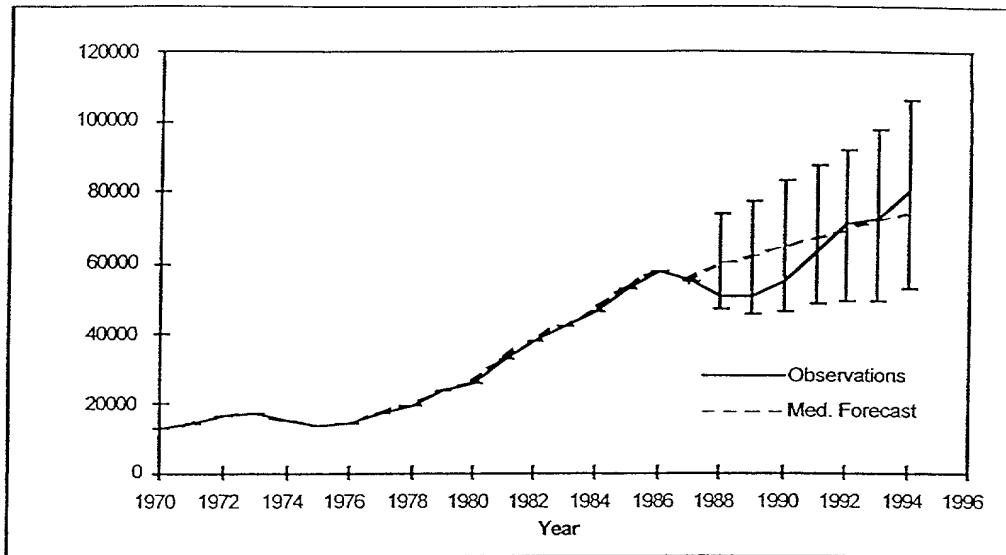


Figure 6.9: Forecast of Rural Commercial Sales and Lower and Upper 95% Confidence Bands in MWH.

Figure 6.9 and Table 6.3 show the conditional forecast of energy sales in the City of Bryan rural commercial class, the forecast errors, and the upper and lower 95% confidence bands on the forecasts. The uncertainty in the energy sales forecast is due to exogenous variables as well as model parameter uncertainties. The average forecast error is 10.8% for the 1988-1994 period. The maximum forecast error was 20.7% in the year 1989.

The uncertainty in future energy sales as represented by the width of the 95% confidence band increases as the forecast moves into the future. This agrees with the intuition. Table 6.3 shows the percentage difference of the lower and upper 95% confidence band limits with respect to the mean forecast. The confidence bands are not symmetric around the mean forecast. The mean forecast is closer to the lower limit of the confidence band throughout the forecasting period. In the year 1988, the upper and the

932

lower limits of the 95% confidence band are 24.3% and 19.5% above and below the mean forecast, respectively. In the year 1988, the 95% confidence band on the energy

932

Table 6.3: Forecast of Rural Commercial Class Sales, Forecast Error and 95% Lower and Upper Limit Confidence Bands in MWH.

Fiscal Year	Sales (MWH)	Forecast and Forecast Error of Energy sales (MWH)					
1970	12615						
1971	13971						
1972	15880						
1973	17014						
1974	14351						
1975	13678						
1976	13833						
1977	16970						
1978	19471						
1979	23715						
1980	25570						
1981	32579						
1982	38280						
1983	42520						
1984	46954						
1985	53437						
1986	57846	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1987	54091						
1988	50381	59079	17.3	47567	73462	19.5	24.3
1989	50823	61343	20.7	45875	77629	25.2	26.5
1990	54420	64050	17.7	46765	83466	27.0	30.3
1991	63111	66545	5.4	48387	87510	27.3	31.5
1992	71020	69137	-2.7	49361	91976	28.6	33.0
1993	72903	71469	-2.0	49545	97466	30.7	36.4
1994	80364	73919	-8.0	53235	105901	28.0	43.3
			10.8				

sales amounts to 43.8% of the average forecast. This percentage grows through the forecasting period. In the year 1994, the upper and lower limits of the 95% confidence band are 43.3% and 28.0% above and below the mean forecast, respectively, and the 95% forecast confidence band on the energy demand amounts to 71% of the sales forecast in the same year.

Above discussion shows that the forecast of energy sales in the city commercial class includes a large level of uncertainty. The source of this uncertainty is contributed to the forecasting model parameters.

VI.3.4 City of Bryan Total System and Peak Load Demand Forecast Uncertainty

In addition to the above four class of customers, the City of Bryan includes schools district which due to its small contribution to total system demand was not modeled separately. The historical information shows that about 6.4% of the total energy sales in the City of Bryan is sold to schools district. The total sales in the City of Bryan is sum of sales to the city division, rural division and schools district. The total system demand includes total sales in the service area and the total system loss. Historically, losses have accounted for about 5.5% of the total system demand in the City of Bryan. To forecasts the total system demand, losses are added to the total sales in the City of Bryan.

Figure 6.10 and Table 6.4 show the historical time series and the conditional forecast of the total system demand in City of Bryan in MWH, the forecast errors, and the upper and lower 95% confidence bands on the forecasts. The uncertainty in the total system demand is due to conditional as well as model parameters. The average forecast error in the forecasting period is 2.9% for the years 1988-1994 period. The maximum forecast error is 4.5% in year 1989.

935

Clearly, the actual system demand is within the 95% confidence bands of the forecasted system demand. The uncertainty in future system demand as represented by

935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

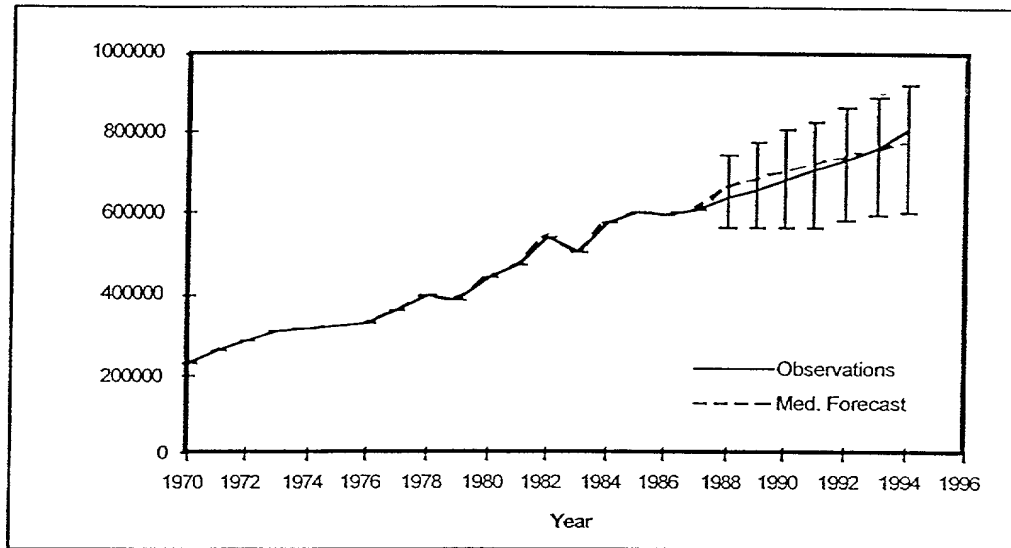


Figure 6.10: Forecast of City of Bryan Total System Demand (MWH) and 95% Confidence Bands.

Table 6.4: Forecast of City of Bryan Total System Demand, Forecast Error and 95% Lower and Upper Confidence Bands in MWH.

Fiscal Year	Demand (MW)	Forecast and Forecast Error of Total System Demand (MWH)					
1970	227707						
1971	258185						
1972	281236						
1973	313221						
1974	316530						
1975	321574						
1976	332385						
1977	358127						
1978	398450						
1979	388358						
1980	436401						
1981	466706						
1982	537147						
1983	499214						
1984	577257						
1985	599182						
1986	594905						
1987	609002	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1988	636682	663147	4.2	97185	79838	14.7	12.0
1989	654049	683591	4.5	116239	95098	17.0	13.9
1990	686083	702572	2.4	137209	106856	19.5	15.2
1991	709782	722241	1.8	159732	106572	22.1	14.8
1992	733681	741284	1.0	158300	125086	21.4	16.9
1993	766369	760353	-0.8	167297	131658	22.0	17.3
1994	808719	779268	-3.6	175183	145230	22.5	18.6
			2.9				

the width of the 95% confidence bands increases as the forecast moves into the future. Table 6.4 shows the percentage difference of the lower and upper 95% confidence bands limits with respect to the mean forecast. The confidence bands are not symmetric around the mean forecast. The mean forecast is closer to the upper limit of the confidence band throughout the forecasting period. In the year 1988, the upper and the lower limits of the 95% confidence bands are 12.0% and 14.7% above and below the mean forecast, respectively. In the year 1988, the 95% confidence bands on the total system demand represents 26.7% of the mean forecast. This percentage grows through the forecasting period. In the year 1994, the upper and lower limits of the 95% confidence bands are 18.6% and 22.5% above and below the mean forecast, respectively. The 95% forecasted confidence band on the total system demand represents 41% of the sales mean forecast in that year.

Clearly, the future forecast of total system demand is uncertain. This level of uncertainty is due to uncertainty in future forecast of exogenous variables and forecasting model parameters.

In modeling the City of Bryan peak load demand an econometric model was developed. After examining several explanatory (exogenous) variables, three variables were selected. these are time, the sales to the city residential class and the CDDs.

Figure 6.11 and Table 6.5 show the historical time series and the conditional forecast of peak load demand in City of Bryan in MW, the forecast errors, and the upper and lower 95% confidence bands on the forecast. The uncertainty in peak load demand is due to uncertainty in exogenous variables as well as uncertainty in forecasting model parameters. The average forecast error is 5.2% for the 1988-1994 period. The maximum forecast error is 8.6% in year 1993.

Clearly, the actual peak load demand is within the 95% confidence bands of the forecasted system demand. The uncertainty in the future peak load demand as

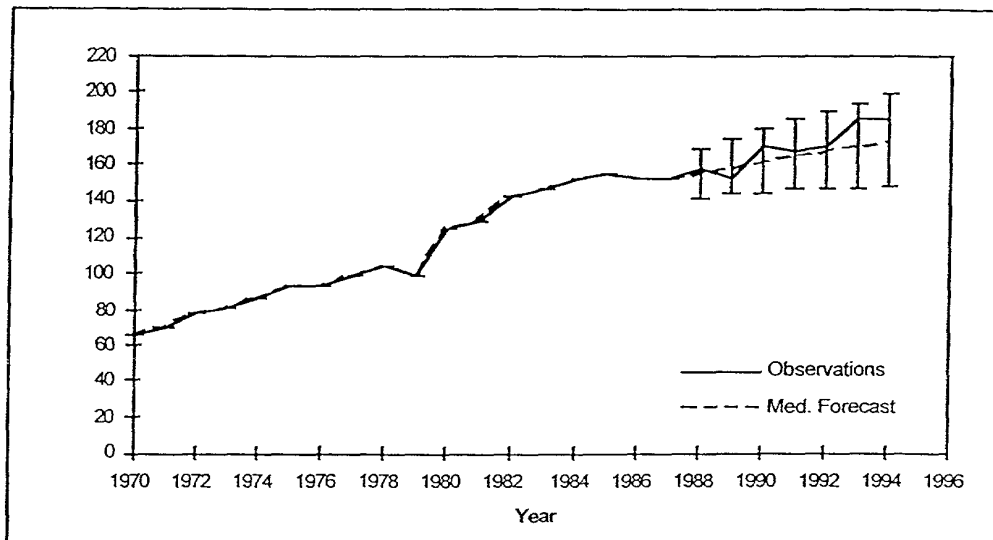


Figure 6.11: Forecast of City of Bryan Peak Load Demand (MW) and 95% Confidence Bands.

represented by the width of the 95% confidence bands increases as the forecast moves into the future. Table 6.5 shows the percentage difference of the lower and upper 95% confidence band limits with respect to the mean forecast. The confidence bands are not symmetric in the years 1989 to 1994. The mean forecast is closer to the lower limit of the confidence band in the year 1989 to 1994. In the year 1988, the upper and the lower limits of the 95% confidence bands are 8.4% above and below the mean forecast of the peak load demand. In the year 1988, the 95% confidence bands on the peak load demand amounts to 16.8% of the mean forecast. This percentage grows through the forecasting period. In the year 1994, the upper and lower limits of the 95% confidence band are 15.0% and 14.5% above and below the mean forecast, respectively. In the year 1994, the 95% confidence band on the peak load demand amounts to 29.5% of the peak load demand forecast in that year.

940

Clearly, the uncertainty in the forecast of the peak load demand is not as large as those forecasted in the future energy sales.

[illegible]

Table 6.5: Forecast of City of Bryan Peak Load Demand, Forecast Error and 95% Lower and Upper Confidence Bands in MW.

Fiscal Year	Peak (MW)	Forecast and Forecast Error of Peak Load Demand(MW)					
1970	66						
1971	70						
1972	78						
1973	81						
1974	87						
1975	93						
1976	93						
1977	99						
1978	104						
1979	99						
1980	125						
1981	129						
1982	143						
1983	147						
1984	152						
1985	155						
1986	152						
1987	152	Base forecast	Error (%)	Lower Limit (95%)	Upper Limit (95%)	LL/Base (%)	UL/Base (%)
1988	158	155	-1.9	142	168	8.4	8.4
1989	153	158	3.3	145	175	8.2	10.8
1990	170	161	-5.3	145	180	9.9	11.8
1991	167	164	-1.8	147	186	10.4	13.4
1992	170	167	-1.8	147	190	12.0	13.8
1993	186	170	-8.6	147	194	13.5	14.1
1994	186	173	-7	148	199	14.5	15.0
			5.2				

VI.4 Discussion of Forecast Uncertainty Models for City of Bryan

In the city commercial class, the actual number of customers and the price of electricity were outside of the 95% uncertainty bands on the forecasts of these exogenous variables. However, the actual sales fell within the 95% confidence band on the forecasted sales.

In the rural residential class, the actual number of customers was within the 95% uncertainty band of the forecasted number. However, the actual price of electricity was outside of the 95% band of the forecasted price. The actual sales in the forecasting period were well within the 95% confidence band of the forecasted sales in this class.

In the rural commercial class, the actual number of customers and the price of electricity were outside of 95% uncertainty bands of the forecasted values. However, the actual sales in the class were within the 95% confidence band of the forecasted sales.

The actual values of total system demand and peak load demand were within the 95% confidence bands of the forecasted values.

A general observation is that while the actual exogenous variables were outside of the 95% uncertainty bands of the forecasted values in many cases, the actual values of energy sales, total system demand and peak load demand were within the 95% confidence bands of the forecasted values. Therefore, large errors in the forecasts of exogenous variables do not result in corresponding large errors in energy sales and demand forecasts using neural network models.

VI.5 Chapter Summary

In this chapter, uncertainty modeling of different classes of customers in the City of Bryan using bootstrapping techniques was presented.

Bootstrap methods do not assume the forecasting model residuals to follow a parametric distribution. In developing the uncertainty models for each class of customer, a forecasting model among the top performing forecasting models is selected. The residual of this forecasting model in the historical period generated the original residual set. Bootstrap residual sets formed through selection by substitution from the original residual set and added to the model prediction in the historical period generated bootstrap realizations of the historical energy sales of peak demand. The bootstrap models were generated using the bootstrap realizations. Using the bootstrap developed models and the distribution of the exogenous variables in the forecasting period, an ensemble of forecasts of energy sales in the forecasting period was generated. The variability in the ensemble of forecasts were presented as an assessment of the uncertainty in the forecasted sales in a the particular class of customers.

The uncertainty in the forecast of energy sales in different classes of customers in City of Bryan was modeled using the above method. The developed uncertainty bands accounted for uncertainties in model exogenous variables and forecasting model parameters. Using the uncertainty in forecasted energy sales, uncertainty in the total system demand was modeled. Moreover, the uncertainty in the forecasted peak load demand was modeled using the above method.

In uncertainty modeling of four classes of customers in the City of Bryan, it was observed that while the actual exogenous variables were outside of 95% confidence bands of their corresponding forecasted values, the actual sales were within 95% confidence

bands of the forecasted energy sales. This shows that large errors in the forecasts of exogenous variables do not result in corresponding large errors in energy sales and demand forecasts using the neural network models in the bootstrapping framework. This is an important goal in the development of any planning process.

— — — — —

CHAPTER VII

SUMMARY AND CONCLUSIONS

VII.1 Summary of the Research Study

The objective of this research was to develop and demonstrate the feasibility of application of several tools to improve long-term load forecasting practice in the electric power industry. Specifically, this research aimed at improvements in two areas: 1) development and demonstration of application of Artificial Neural Networks (ANN) in generating more accurate long-term forecasts of energy consumption and peak load demand and 2) development of quantitative methods for assessing uncertainty in forecasts. The long-term load forecasting methods developed were tested by forecasting the energy demands for different classes of customers and for total system energy demand and total peak load demands in the member utilities of the Texas Municipally Power Agency (TMPA). Forecasts using the ANN models developed were shown to outperform the currently available forecasting methods in the utilities studied. The quantitative uncertainty assessment methods developed were applied to quantify the uncertainty level in long-term forecasts of energy sales and peak load demand in the City of Bryan.

A first step in the utility planning process is forecasting long-term energy sales and peak load demand. Other components of the utility power system planning process are then based on the results of long-term load forecasts. Thus, the accuracy of long-term load forecasts is a critical issue. An integrated long-term load forecasting system should include a module to develop point forecasts of energy demand as well as an uncertainty assessment module to quantify the uncertainty of forecasts arising from uncertainties in input and model parameters. Over the last two decades there have been many attempts

to improve long-term load forecasting methods and many different modeling techniques have been used including linear and nonlinear time series methods, end-use methods and econometric and hybrid methods. However, the ever increasing complexity of the electric utility industry due to factors such as technological and economic change and changes in conservation and deregulation policies as well as the large financial incentives for more accurate forecasts have kept the search for more accurate forecasting methods alive.

The methods developed in this research can be classified as econometric methods. Econometric methods attempt to identify the cause and effect relationship between different influencing social, economic and geographic variables and annual MWH energy sales or MW peak load demand. Adopting this method, the forecaster tries to accurately identify the model structure explaining this relationship and the parameters of the particular model structure. The accuracy of any long-term load forecast relies primarily on accurate identification of the forecasting model structure and its parameters.

Chapter II presented a system view of the load forecasting problem. Identifying the forecasting model as a system with appropriate inputs permits the application of a vast number of methods in the field of system identification to this problem. A system model is a mathematical model which relates changes in the system inputs or stimuli to the model output, energy sales or peak load demand in this case. A system model can be linear or nonlinear. Linear system models assume the relationship between inputs and outputs can be expressed in a linear form. The most common type of linear input-output system models are the AutoRegressive with eXogenous input (ARX) and the Auto-Regressive Moving-Average with eXogenous input (ARMAX) models. The single-step-ahead prediction (SSP) and multi-step-ahead prediction (MSP) forms of input-output models were described. It was demonstrated that MSP forms of such models can be recursively obtained from the SSP forms by replacing the output observations with the past predictions of the model in the forecasting period.

The most common nonlinear input-output system models are the Nonlinear AutoRegressive with eXogenous input (NARX) and Nonlinear Auto-Regressive Moving-Average with eXogenous input (NARMAX) models, which are the nonlinear analogs of the ARX and ARMAX models. However, unlike ARX and ARMAX, the relationship between inputs and outputs of the NARX and NARMAX models are nonlinear. The SSP and MSP forms of NARX and NARMAX were presented. The polynomial and Neural Network (NN) expansions of NARX were reviewed. Using a polynomial expansion of NARX, a system model of m -input, n -output form is transformed into n models of m -input single output system models.

The Neural Network (NN) family of models such as the Feedforward Multilayer Perceptron (FMLP) are nonlinear model structures that have been shown effective in modeling nonlinear systems. Further, in addition to being nonlinear, the aforementioned NN is a nonparametric model structure. The FMLP can be shown to belong to be a class of NARX system models. They have been shown to exhibit superior performance compared to other NARX type system models, especially when MSP performance is the evaluation criteria. Finally, fairly systematic methods exist that can identify parameters of NN system models. In NN terminology, the NN parameter estimation process is referred to as training and model parameters are referred to as weights and biases of the NN model. Each building block of a NN is called a neuron. Each NN model structure consists of at least three layers of neurons: the input layer, the hidden layer and the output layer. In FMLP family of NN model structures, each neuron receives input only from neurons in the previous layer. The neuron output is a nonlinear transformation of the weighted sum of its inputs.

System Identification refers to systematic procedures followed to develop mathematical models of the system. Considering a particular system model, prediction error methods try to estimate model parameters so to minimize the model prediction error. The most common linear estimation method, linear Least Square Estimation (LSE), was reviewed. The LSE method is based on minimizing the squares of the model

prediction errors in order to determine parameters of linear models such as ARX and ARMAX models. Other linear estimation techniques are Maximum Likelihood Estimation (MLE) and Instrument Variables (IV) method.

The most widely used method for training an FMLP is the Backpropagation (BP) algorithm and its variants. The BP algorithm is basically a LSE algorithm that attempts to minimize an error criterion by adjusting the weights of the network. At every iteration each input sample is presented to the input layer. The NN predicts the output and its difference with the observed output is computed. This error is propagated back through the network and its gradient with respect to each model parameter (weight) is computed. The adjustment in each weight is in proportion to the gradient of the output error with respect to the particular weight value. Adjusting weights through a large number of iterations causes weight values to converge to values which cause the network outputs to be close to the observed outputs. This process is called training. However, the training process in its basic form is not adequate to deal with the unique characteristics of data sets found in long-term load forecasting.

Chapter III presented a nonlinear view of econometric long-term load forecasting models. The different characteristics of the long-term load forecasting problem such as cause-and-effect relationships between socio-economic indicators and energy sales, the time series correlation between energy sales in consecutive years, and nonlinear effect between particular exogenous variables and energy sales were discussed.

The available historical data set in long-term load forecasting is short. Available historical data spans at most 30 years of annual data. Also, data from the distant past may not be representative of the present or future for a variety of reasons including technological change and social and economic change. Further, historical data is often corrupted by changes in utility service territories, by changes in rate schedules and customer classifications, and by change in data collection definitions and procedures.

Therefore, historical data sets, the basis of forecasting models, tend to contain errors of various kinds.

Traditionally, all of the historical data set has been used in developing a forecasting model. We proposed to reserve a portion of the historical data set for final evaluation of developed forecasting models. This portion is called the forecasting (or backcasting) set. The remaining portion is called the estimation set. Moreover, the estimation set is divided into two subsets, the training set and the validation set. The training set is directly used in model parameter estimation and the validation set is used to prevent overfitting and underfitting.

Several estimation techniques were proposed to deal with the training and the testing set. The advantages and drawbacks of each method in dealing with long-term load forecasting data sets were discussed. Models developed based on the performance on the validation set only suffer from underfitting of the training set and tend to give poor results in forecasting future loads. Models developed based on selection of a particular error level in the training set require the forecaster to find an error tolerance level in the training set which results in good performance on the validation set. Additional observations regarding the difficulty of dealing with long-term load forecasting set is the fact that model with number of parameters such as different hidden layers, hidden nodes, autocorrelation and crosscorrelation terms result in different forecasting performance. An reliable forecasting system should take into consideration the above observations and optimize the forecasting model parameters to result in best forecasting models. To optimize these parameters the application of Monte Carlo filtering process was proposed.

In applying Monte Carlo filtering process a number of acceptable behaviors and variables are selected. The set of acceptable behaviors include crossing of the training error and the validation error profiles and the error level at the crossing point. The set of variables include the number of hidden layers and nodes and the autocorrelation and

cross-correlation terms and model parameters. Through application of Monte Carlo filtering process a number of parameter combinations which satisfy the acceptable behaviors are selected.

The proposed Monte Carlo filtering process includes a deterministic and a stochastic search component. The deterministic component selects the best model structure. This search is confined to one hidden layer, a small number of hidden nodes, and a few autocorrelation and cross-correlation terms. The stochastic component searches model parameters (weights) for each combination of the structure parameters to identify models satisfying the acceptable behavior conditions. It is generally necessary to perform several hundred weight initializations for each model structure to obtain a few models which satisfy acceptable behavior conditions. The result of the stochastic search is a predetermined number of networks with the lowest crossing error levels for the training and validation sets for the particular model structure. Moreover, as a performance index of the forecasting capability of this model structure, the mean and the variance of the error level of this predetermined number of networks is computed.

The above stochastic search is performed for different model structures set in the deterministic Monte Carlo filtering search. A forecasting performance index is computed for each model structure, based on the mean error and the variance of the selected models in the particular model structure. Selection of the best model structures is based on these forecasting performance indices. A number of the model structures with the highest forecasting performance index are selected for final energy sales or peak load demand forecast. These top-performing model structures are used to forecast energy sales in the forecasting period. The final forecast is the average of forecasts of these top-performing forecasting models.

Chapter IV presented the application of the proposed nonlinear forecasting methods developed in Chapter III to forecast annual energy sales and peak load demand in the member utilities of the Texas Municipal Power Agency (TMPA), namely City of

Bryan, City of Denton, City of Garland and City of Greenville. The historical information and other critical expertise were provided by the member utilities. A portion of the historical data set was reserved for final evaluation and comparison. This portion of the historical data set was called the forecasting period. The length of the forecasting period was different in the four utilities. For the City of Greenville, due to an extremely short historical time series, only a total system model for energy sales was developed. Both conditional and unconditional forecasts of the sales were developed. For those utilities which provided a reference forecast made sometime in past, an attempt was made to compare the accuracy of the forecasts made by the utilities and those made using the methods developed here.

Two nonlinear forecasting methods were developed. In the NN-1 forecasting method, the stopping criterion given by equation 3.13 was used. Using this stopping criterion, the training set is trained to a prespecified error level and the performance of the obtained model on the validation set is evaluated. Models with low errors on the validation set are selected as satisfying the acceptable behavior conditions. In the NN-2 forecasting method, the stopping criterion given by equation 3.14 was used. Using this stopping criterion, models in which the training set and the validation set error profiles cross during the training process are selected as satisfying the acceptable behavior conditions.

Eighteen different forecasting models for annual energy sales and total system demand for the four utilities were developed. In addition four forecasting models for annual peak load demand were developed using NN-1 and NN-2 forecasting methods. The conditional and unconditional forecasts during the forecasting period were developed.

With exception of two classes of customers, NN-1 and NN-2 exhibited improved accuracy as compared to utility-provided forecasts when considering forecasted exogenous variables. With the exception of one energy sales model, the average

conditional forecast error for NN-1 and NN-2 forecasting methods were less than 11% and 13.4%, respectively. The average conditional peak load demand forecast error for NN-1 and NN-2 forecasting methods were 11.8% and 8.9%, respectively. Both NN forecasting methods outperformed the utility provided conditional forecasts. NN-2 outperformed NN-1 in conditional annual energy sales forecast. However, NN-1 outperformed NN-2 in annual peak load demand forecast.

One of the major sources of inefficiency in the electric utility industry is uncertainty and associated risk with future energy sales and peak load demands. Inability to accurately forecast future trends and changes affecting long-term customer demand can easily translate into major economic penalties for utilities involved and as a consequence for the various customers being served. While recognizing the good progress in the accuracy improvement of long-term load forecasting methods in the last two decades, we still have to recognize the fact that *The Forecast Is Always Wrong!*. Long-term forecasts of energy sales and total system and peak load demand are based on forecasts of many variables well into the future. These forecasts include a high level of uncertainty. Moreover, due to the unique characteristics of historical data sets, the developed forecasting models exhibit a large level of uncertainty in model structure and in model parameters. Therefore, an essential component of an integrated long-term load forecasting system is a quantification of the uncertainty associated with the forecasts. The outcome of this assessment is represented as confidence bands around the point forecasts.

Several methods have been used in predicting confidence bands around point forecasts. However, they are based on unrealistic assumptions such as accurate identification of the forecasting model or the availability of infinite historical observations. Using these assumptions, the uncertainty in future sales is obtained by passing the uncertainty in future exogenous variables through the forecasting model. However, lack of confidence in such simplistic procedures has resulted in utilities adopting empirical estimates of uncertainty in future energy forecasts.

Chapter V presented resampling techniques as methods to compute the uncertainty in the estimate of a particular statistic. In particular, the bootstrapping method was reviewed in detail as a method to quantify uncertainty in model parameters. Having the data at hand as the only realization of the original distribution, bootstrap repeatedly uses it to generate an ensemble of realizations. The variability of the original estimate in this ensemble is an indication of the variability of the original estimate in the original distribution.

Bootstrap relies on the repeated use of a point forecaster to generate the future forecast ensemble. Therefore, it is important to investigate the effect of a particular point forecaster on the final confidence bands. Three different point forecasters were developed, namely the ordinary linear regression forecaster, the regularized linear regression and the regularized neural network forecaster. The ordinary regression model uses all the estimation set directly in parameter estimation. The regularized regression method and regularized NN method divide the estimation set into a training set and a validation set. The model parameter estimation stopping criterion was defined to be the crossing of the training set and the validation set error profiles. In the regularized regression and regularized NN forecasters, a forecasting model among the top-performing forecasting models was selected and its corresponding historical prediction was used to generate the original bootstrap residuals. Repeated selection from this residual set and addition to the original model prediction in the historical period generated bootstrap historical observations. The combination of historical exogenous variables and bootstrap realizations of energy sales generated bootstrap historical sets. A model was developed for each bootstrap historical set. A similar procedure to that used in generating the original point forecast was used in modeling each bootstrap realization.

The future distribution of exogenous variables can be passed through each bootstrap developed model to generate a forecast distribution of future energy sales. Repeating this process several hundred times generates an ensemble of future sales realizations. The variability in this ensemble is a measure of uncertainty in future energy

sales due to model parameter and input variable uncertainties. A nonparametric confidence interval computation method was used to calculate 95% confidence bands.

Application of the above three point forecasters in assessing the uncertainty in the future sales in the city residential class of City of Bryan showed that the three methods generated 95% confidence bands which bounded actual sales in the forecasting period. Ordinary linear regression method resulted in the lowest average point forecast error and the narrowest 95% confidence band in the forecasting period. The regularized linear regression method resulted in the highest average point forecast error and the widest 95% confidence band in the forecasting period. This intent of this study was to investigate the effects of point forecast on forecast uncertainty. It was shown in chapter IV that the regularized neural network models outperformed ordinary regression models in forecasting the energy sales and peak load demand in the member utilities of TMPA in 85% of the cases studied. The reported ordinary regression model was developed in the conduct of this research using a different forecasting than that provided by the City of Bryan. It can be expected that if another model had been selected for this study, the results of point forecast could have been more favorable towards the regularized neural network forecaster. The only conclusion from this study is that the regularized neural network models do not necessarily generate the narrowest confidence bands around the forecasts.

Generally, the accuracy of the point forecast is referred to as the forecast bias and the width of the forecast confidence band is referred to as the forecast variance. This study showed that minimizing the forecast bias does not necessarily result in minimizing the forecast variance. The forecast bias is related to the bias in the historical period through a non-monotonic function. The variance of the forecast is also related to the variance of the forecasting model in the historical period. In this research study, we developed a nonlinear forecasting method to minimize the bias of the forecast. That is, we developed methods for more accurate forecasts. However, to reach a lower forecast bias, we increased the model bias in the historical period. This in turn resulted in an

increase in the model variance in the historical period. An increase in the model variance in the historical period, however, results in an increase in the forecast variance. As a result, the confidence bands generated using the regularized neural network models tend to be larger than those generated by the ordinary regression models. Nevertheless, in general, the former provides more accurate point forecasts from the later and it is the preferred long-term load forecasting method.

The above discussion presents the dilemma regarding forecast bias and forecast variance. Decreasing the forecast bias does not necessarily result in a decrease in the forecast variance or in another words narrower confidence bands.

Chapter VI demonstrated the application of the developed uncertainty assessment methods to different customer classes in City of Bryan. To reliably compute 95% confidence bands around sales forecasts, several hundred bootstrap realizations were generated. Future distribution of the exogenous variables were passed through each bootstrap developed models to account for both model parameter and input parameters uncertainty. A nonparametric CI computation method was used to compute 95% confidence bands of the future sales around the point forecast. The width of 95% confidence band for different class of customers were different. This width for the rural commercial class reached as much as 70% of forecast of sales in a 7 year forecast. The 95% confidence band of total system demand was computed using the uncertainty in sales forecasts of difference classes of customers. The width of 95% confidence band around the total system demand reached 40% of total system demand in 7 year forecast. Finally, an uncertainty model was developed for peak load demand in the City of Bryan. The 95% confidence bands around the forecast of the peak load demand was determined to be 30% of the peak demand forecast in the 7 year forecast.

It was observed that while the actual exogenous variables were outside of the 95% uncertainty band of their corresponding forecasted values in many cases, the actual values of energy sales, total system and peak load demand were within the 95%

confidence bands of the forecasted values. Therefore, errors in forecasts of exogenous variables did not result in corresponding large errors in energy sales and demand forecasts.

In summary, the application of the developed nonlinear forecasting methods to forecasting the sales and peak load demand in member utilities of TMPA, and uncertainty assessment of the forecasted sales in different classes of customers in the City of Bryan present enough evidence to consider the proposed methods as viable tools in long-term load forecasting practice.

VII.2 Conclusions from the Research Study

The objective of this research study has been to develop a comprehensive long-term load forecasting system which includes an accurate and robust long-term load forecaster and a stochastic uncertainty assessment technique. The objective was achieved by using recent advances in the fields of system identification, artificial neural networks, and statistical resampling methods.

The conclusions drawn from this research study can be summarized as follows:

1. Data scarcity imposes restrictions on effective solutions of the long-term load forecasting problem.
2. In data scarce forecasting problems, nonlinear estimators of controlled complexity are viable alternatives to linear estimators.
3. In long-term load forecasting, Monte Carlo filtering is a viable approach to objectively establish forecast model structure and parameter values, and improve confidence in model predictions.

4. Combination of long-term forecasts of different forecasting models generates more accurate forecasts than long-term forecasts from a single forecasting model.
5. The bootstrapping method can be applied to compute the uncertainty in future annual energy sales and peak load demand.
6. There is no well-defined relation between the forecast bias and the forecast variance.

VII.3 Recommendations for Future Research Directions

The nonlinear forecasting methods and uncertainty assessment techniques developed in this research are only the starting point for further investigation into application of Monte Carlo filtering process, nonlinear and statistical bootstrapping methods in long-term load forecasting practice. Based on the research reported in this dissertation, some possible topics for future research are:

1. Investigation of selection of the stopping criterion to generate more accurate long-term load forecasts
2. More intelligent selection for the range of parameters in Monte Carlo filtering process.
3. More intelligent combination of the forecasts generated from individual forecasting models.
4. Investigation of selection of a residual set among residual sets of different forecasting models.
5. Investigation of the forecast “bias” and “variance” dilemma.

REFERENCES

- [1] E. Barakat, J. Al-Qassim and S. Al-Rashed, "New Model for Peak Demand Forecasting Applied to Highly Complex Characteristics of a Fast Developing Area", *IEE Proc.-C*, Vol. 139, No. 2, pp. 136-140, 1992.
- [2] E. Barakat and S. Al-Rashed, "Long Range Peak Demand Forecasting Under Conditions of High Growth", *IEEE Transactions on Power Systems*, Vol. 7, No. 4, pp. 1483-1486, 1992.
- [3] A. Barron, "Universal Approximation Bounds for Superpositions of a Sigmoid Function", *IEEE Transactions on Information Theory*, Vol. 93, pp. 930-945, 1993.
- [4] D. Bates, and D. Watts, *Nonlinear Regression Analysis and Its Applications*, New York: Wiley, 1988.
- [5] W. Bentley, C. Cosgrove and P. Stallcup, "Integrating Econometric and End-Use Models: A Realistic Approach to Conservation Programs", in *Proc. 3rd EPRI Load-Forecasting Symposium*, 1982, pp. 44-76.
- [6] J. Bernard and M. Veall, "The Probability Distribution of Future Demand: The Case of Hydro Quebec", *Journal of Business and Economic Statistics*, pp. 417-424, July 1987.
- [7] T. Bhattacharaya and T. Basu, "Medium Range Forecasting of Power System Load Using Modified Kalman Filter and Walsh Transform", *International Journal of Power Systems*, Vol. 15, No. 2, pp. 109-115, 1993.
- [8] S. Billings, S. Chen and A. Korenberg, "Identification of MIMO Nonlinear Systems Using a Forward-Regression Orthogonal Estimator", *International Journal of Control*, Vol. 49, pp. 2157-2189, 1989.
- [9] S. Billings, H. Jamaluddin and S. Chen, "Properties of Neural Networks with Applications to Modelling Nonlinear Dynamic Systems", *International Journal of Control*, Vol. 55, pp. 193-224, 1992.
- [10] R. Billinton and R. Allan, "Power System Reliability in Prospective", *IEE Journal of Electronics and Power*, Vol. 30, pp. 231-236, March 1984.

- [11] D. Bonney, "Pacific Gas and Electric Company Residential End-Use Data Development", in *Proc. 3rd EPRI Load-Forecasting Symposium*, 1982, pp. 21-28.
- [12] W. Bruke, F. Schweppe, B. Bruke, "Trade Off Methods in System Planning", *IEEE Transactions on Power Systems*, Vol. 3, No. 3, pp. 1284-1290, 1988.
- [13] Bryan Utility Staff, *City of Bryan, Texas, System Load Forecast*, Bryan, TX: City of Bryan, 1984.
- [14] S. Buckland, "Calculation of Monte Carlo Confidence Intervals", *Applied Statistics*, Vol. 34, pp. 296-301, 1985.
- [15] S. Buckland, "Monte Carlo Confidence Intervals Estimation Using the Bootstrap Technique", *BIAS*, Vol. 10, pp. 194-212, 1983.
- [16] K. Chakraborty, K. Mehrotra, C. Mohan and S. Ranka, "Forecasting the Behavior of Multivariate Time Series Using Neural Networks," *Neural Networks*, Vol. 5, pp. 961-970, 1992.
- [17] C. Chatfield, "Calculating Interval Forecasts", *Journal of Business and Economic Statistics*, Vol. 11, No. 2, pp. 121-135, 1993.
- [18] S. Chen and S. Billings, "Representations of Nonlinear Systems: the ARMAX Model", *International Journal of Control*, Vol. 49, pp. 569-594, 1988.
- [19] S. Chen, S. Billings and P. Grant, "Nonlinear System Identification Using Neural Networks", *International Journal of Control*, Vol. 51, pp. 1191-1214, 1990.
- [20] Z. Chen and X. Gang, "Long-Term Load Forecasting for Pudong New Area of Shanghai Using Grey Theory", *Automation of Electric Power Systems*, Vol. 17, No. 7, pp. 20-24, 1993.
- [21] E. Crooke, "The Forecast's Role in Decision Making", in *Proc. 8th Electric Utility Forecasting Symposium*, 1991, pp. 2-1,2-6.
- [22] Denton Utility staff, *City of Denton, Texas, System Load Forecast*, Denton, TX: City of Denton, 1992.
- [23] Edison Electric Institute, *Electric Forecasting Methodology*, Boston, MA: Charles River Associates, 1986.

- [24] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," *Annals of Statistics*, Vol. 7, No. 1, pp.1-26, 1979.
- [25] B. Efron, "Computers and the Theory of Statistics: Thinking the Unthinkable", *SIAM Reviews*, Vol. 2, No. 4, pp. 460-479, 1979.
- [26] B. Efron and Gail Gong, "A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation", *The American Statistician*, Vol. 37, No. 1, pp. 36-48, 1983.
- [27] B. Efron and R. Tibshirani, "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy", *Statistical Science*, Vol. 1, No. 1, pp. 54-77, 1986.
- [28] Electric Power Research Institute, *Uncertainty in Forecasting*, Oakland, CA: Barakat and Chamberlin Inc., 1990.
- [29] Electric Power Research Institute, *Practical Applications of Forecasting Under Uncertainty*, Oakland, CA: Barakat and Chamberlin Inc., 1992.
- [30] D. Freedman, "Bootstrapping Regression Models", *Annals of Statistics*, Vol. 9, pp. 1218-1228, 1981.
- [31] D. Freedman and S. Peters, "Bootstrapping a Regression Equation: Some Empirical Results", *Journal of American Statistical Association*, Vol. 79, pp. 97-106, March 1984.
- [32] D. Freedman and S. Peters, "Using the Bootstrap to Evaluate Forecasting Equations", *Journal of Forecasting*, Vol. 4, pp. 251-262, 1985.
- [33] X. Gang, "Long-Term Forecasting of Electric Power Demand Using Grey Theory", *Journal of Shanghai Jiaotong University*, Vol. 27, No. 5, pp. 98-103, 1993.
- [34] R. Gardner, "A Unified Approach to Sensitivity and Uncertainty", in *Proc. 10th IASTED International Symposium* 1984, pp. 155-157.
- [35] C. Grainger and P. Newbold, "Improved Methods of Combining Forecasts", *Journal of Forecasting*, Vol. 3, pp. 197-204, 1984.
- [36] P. Henault, R. Eastvedt, J. Peschon and L. Hajdu, "Power System Long-Term Planning in the Presence of Uncertainty", *IEEE Trans. on PAS-89*, No. 1, pp.156-164, 1970.

- [37] G. Hornberger and B. Cosby, "Selection of Parameter Values in Environmental Models Using Sparse Data", *Applied Mathematics and Computation*, Vol. 17, pp. 335-355, 1985.
- [38] H. Hutsch "The Jackknife and the Bootstrap for General Stationary Observations", *Annals of Statistics*, Vol. 17, pp. 1217-1241, 1989.
- [39] W. Irish, "A Probabilistic Approach to Evaluating Load Forecast Uncertainty", in *Proc. 8th Electric Utility Forecasting Symposium* 1991, pp. 45-1, 45-16.
- [40] D. Kexel, "The Ten Year Forecast: How Certain Are We", in *Proc. 8th Electric Utility Forecasting Symposium*, 1991, pp. 44-1, 44-14.
- [41] H. Klein and J. Vetter, "Designing Scenarios for Planning and Decision-Making: A Demonstration of the Spire Approach", in *Proc. 8th Electric Utility Forecasting Symposium*, 1991, pp. 50-1, 50-12.
- [42] R. Lepage and L. Billard, *Exploring the Limits of Bootstrap*, New York: Wiley, 1992.
- [43] X. Liu, B. Ang and T. Goh, "Forecasting of Electricity Consumption: A Comparison Between an Econometric Model and a Neural Network Model", in *Proc. of IEEE Inter. Joint Conf. on Neural Networks*, 1991, pp. 1254-1259.
- [44] K. Liu, S. Subbarayan, et al., "Comparison of Very-Short Load Forecasting Techniques", *IEEE Transactions on Power Systems*, Vol. 11, No. 2, pp. 877-882, 1996.
- [45] L. Ljung, *System Identification: Theory for Use*, NJ: Prentice-Hall, Hertford-shire, UK, 1987.
- [46] J. McMenamin, "BG&E Statistically Adjusted Residential End-Use Models", in *Proc. 8th Electric Utility Forecasting Symposium* 1991, pp. 40-1, 40-10.
- [47] C. Mooney and R. Duval, *Bootstrapping: A Nonparametric Approach to Statistical Inference*, Newbury Park, CA: Sage Publications, 1995.
- [48] P. Newbold, C. Grainger and R. Ramanathan, "Improved Methods of Combining Forecasts", *Journal of Forecasting*, Vol. 8, pp. 197-204, 1989.
- [49] A. Papalexopolous, S. Hao and T. Peng, "An Implementation of a Neural Network Based Load Forecasting Model for the EMS", *IEEE Transactions on Power Systems*, Vol. 9, No. 4, pp. 1956-1962, 1994.

- [50] A. Parlos, B. Fernandez, A. Atiya, J. Muthusami, and W. Tsai, "An Accelerated Learning Algorithm for Multilayer Perceptron Network", *IEEE Transactions on Neural Networks*, Vol. 5, pp. 493-497, May 1994.
- [51] A. Parlos, E. Oufi, J. Muthusami and A. Patton, *Long-Term Forecasting for Electric Utilities Using Neural Information Processing Systems*, Washington. D.C.: American Public Power Associations, 1994.
- [52] A. D. Patton, "Uncertainty in Future Capacity Need", in *Proc. NSF/EEI Workshop on Research Needs for Coping with Uncertainty in Power Systems*, 1991, pp. 9-15.
- [53] M. Pesaran and L. Slater, *Dynamic Regression: Theory and Algorithms*, London, UK: Ellis Horwood Publishers, 1980.
- [54] S. Peters and D. Freedman, "Using the Bootstrap to Evaluate Forecasting Equation", *Journal of Forecasting*, Vol. 4, pp. 251-262, 1985.
- [55] R. Pindyck and D. Rubinfeld, *Econometric Models and Economic Forecasts*, New York: Mc-Graw Hill, 1991.
- [56] W. Pitts and W. McCulloch, "How We Know Universals: The Perception of Auditory and Visual forms", *Bulletin of Mathematical Biophysics*, Vol. 9, pp.127-147, 1947.
- [57] The Public Utility Commission of Texas, *Long-Term Electric Peak Demand and Capacity Resource Forecast for Texas*, Austin, TX: Public Utility Commission of Texas, 1990.
- [58] The Public Utility Commission of Texas, *Long-Term Electric Peak Demand and Capacity Resource Forecast for Texas*, Austin, TX: Public Utility Commission of Texas, 1993.
- [59] O. Rais, *Modeling Complex Process Systems Using Recurrent Multilayer Perceptron*, Ph.D. Dissertation, Texas A&M University, College Station, TX, 1995.
- [60] K. Rose, E. Smith, R. Gardner, A. Brenkert and S. Bartell, "Parameter Sensitivity, Monte Carlo Filtering, and Model Forecasting Under Uncertainty", *Journal of Forecasting*, Vol. 10, pp. 117-133, 1991.
- [61] D. Rumelhart and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, I & II*, Cambridge, MA: MIT Press, 1986.

- [62] J. Sangregorio and A. Zakem, "Uncertainty in Load Forecasting", in *Proc. 3rd EPRI Load-Forecasting Symposium*, 1981, pp.123-143.
- [63] F. Schweppe, M. Caramanis, R. Tabors, R. Bohn, *Spot Pricing of Electricity*, Boston MA: Kluwer Academic Press, 1988.
- [64] H. Seeyle, "Trend Prediction Important in Planning", *Electrical World*, June 22, pp. 79-81, 1946.
- [65] J. Sjöberg, H. Hjalmarsson and L. Ljung, "Neural Networks in System Identification", in *Proc. 10th IFAC Symposium on System Identification*, Copenhagen, Denmark, 1994, Vol. 2, pp. 49-72.
- [66] J. Sjöberg and L. Ljung, "Criterion Minimization Using Estimation Data and Validation Data", in *Proc. 10th IFAC Symposium on System Identification*, Copenhagen, Denmark, 1994, Vol. 1, pp. 187-190.
- [67] J. Sjöberg and L. Ljung, "Overtraining, Regularization, and Searching for Minimum in Neural Networks", in *Proc. IFAC Symposium on Adaptive Systems in Control and Signal Processing*, Grenoble, France, 1992 pp. 669-674.
- [68] D. Smith, "Combination of Forecasts in Electricity Demand Prediction", *Journal of Forecasting*, Vol. 8, pp. 349-356, 1989.
- [69] K. Stanton and P. Gupta, "Forecasting Annual or Seasonal Peak Demand in Electric Utility System", *IEEE Trans. on PAS-89*, pp. 951-959, 1970.
- [70] H. Stoll, *Least-Cost Electric Utility Planning*, New York: Wiley, 1991.
- [71] V. Strejc, "Least Squares Parameter Estimation", *Automatica*, Vol. 16, pp. 535-550, 1990.
- [72] T. Tummula and G. Fung, "Forecasting of Electricity Consumption: A Comparative Analysis of Regression and Artificial Neural Network Models", in *Proc. IEE 2nd Int. Conf. on Advances in Power System Control, Operation and Management*, 1994, pp. 782-788.
- [73] P. Westfall and S. Young, *Resampling-Based Multiple Testing Examples and Methods for P-value Adjustment*, New York: Wiley Interscience Publications, 1993.

- [74] G. W. Westley, *New Directions in Econometric Modeling of Energy Demand With Applications to Latin America*, New York: Inter-American Development Bank, 1992.
- [75] H. Willis and H. Tran, "Load Forecasting for Transmission Planning", *IEEE Trans. on PAS-103*, No. 3, pp. 561-570, 1984.

965

WHAT IS CLAIMED IS:

1. A system for condition assessment of a piece of equipment comprising:

a virtual condition assessment instrument for
5 measuring a condition of the piece of equipment,
comprising:

a data capture subsystem for sampling a set of
analog signals and converting the set of analog signals
to at least one digital signal;

10 a model-based component, comprising:

a filter to estimate disturbances; and
a predictor for predicting an expected
response;

a signal-based component for processing output
15 from the model-based component;

a classification component for processing
output from the signal-based component;

a fuzzy logic expert component for combining
information from the classification component and the
20 model-based component to assess the condition of the
equipment; and

a condition assessment panel for displaying the
condition of the equipment; and

a virtual end-of-life prediction instrument for
25 measuring an end of life of the piece of equipment,
comprising:

966

a condition prediction end-of-life prediction component for analyzing information from the virtual condition assessment instrument to predict condition and end-of life of the equipment;

5 a prediction condition and end-of-life uncertainty estimation component for processing information received from the condition prediction end-of-life prediction component to obtain an estimate of the uncertainty of the condition and end-of-life prediction;

10 and

an end-of-life panel for displaying the condition and end-of-life prediction and uncertainty.

967

ABSTRACT OF THE DISCLOSURE

A condition assessment and end-of-life prediction system that includes a virtual condition assessment instrument and a virtual end-of-life prediction instrument. The virtual condition assessment instrument measures the condition of the equipment and includes a data capture subsystem for sampling a set of analog signals and converting them into digital signals, a model-based component to estimate disturbances and predict an expected response, a signal-based component to process output from the model-based component, a classification component to process output from the signal-based component, a fuzzy logic expert component to combine information from the classification component and the model-based component in order to assess the condition of the equipment, and a condition assessment panel to display the condition of the equipment. The a virtual end-of-life prediction instrument predicts the equipment end-of-life and includes a condition prediction end-of-life prediction component to analyze information from the virtual condition assessment instrument to predict condition and end-of life, a prediction condition and end-of-life uncertainty estimation component to estimate the uncertainty of the condition and end-of-life prediction, and an end-of-life panel for displaying the condition and end-of-life prediction and uncertainty.

TAMUS 1058

Methods for constructing a general K-step-ahead (single or multi) predictor; methods apply to adaptive and non-adaptive predictor construction. One embodiment of these methods made use of the architecture in US Pat. No. 5,479,571. Implementation of the methods of construction using other architectures (or models) is feasible.

TAMUS 1084

Methods for constructing a general state predictor and state filter; methods are presented for: (1) adaptive, (2) non-adaptive, and (3) hybrid construction. One embodiment of these methods make use of the architecture of US Pat. No. 5,479,571 and methods developed under TAMUS 1058. Implementation of the methods of construction using other architectures (or models) is feasible.

TAMUS 1097

Methods for constructing a general forecasting system and its associated forecasting uncertainty from sparse data sets. Implementation of the methods of construction using any architecture (or model) is feasible.

TAMUS 1059

Methods for construction of general equipment condition assessment (or diagnosis) systems. Also methods for computing the associated diagnosis uncertainty are claimed. Three methods are disclosed: (1) model-based, (2) signal-based, and (3) hybrid (both model-based and signal-based). The embodiment of these methods make use of the architecture in US Pat. No. 5,479,571, and the disclosures TAMUS 1058 and TAMUS 1084. Implementation of the methods of construction using other architectures (or models) is feasible. One such implementation for an AC induction motor is given.

Methods for construction of general equipment end-of-life prediction (prognosis) systems. Also methods for computing the associated prognosis uncertainty are claimed. The embodiment of these methods make use of the architecture in US Pat. No. 5,479,571, and the disclosures TAMUS 1058 and TAMUS 1084. Implementation of the methods of construction using other architectures (or models) is feasible.

TAMUS 1161

A virtual instrument for measuring the condition of equipment in general. An equipment condition instrument displays information regarding present condition of incipient failures, the uncertainty associated with them, equipment efficiency, the cost associated with efficiency degradation, recommended repairs, if any, the associated costs with them, and the cost associated with the downtime needed to perform the repairs. The embodiment of this instrument makes use of the algorithms in disclosure TAMUS 1059. Implementation of the instrument using other algorithms is feasible. One such implementation for an AC induction motor is given.

A virtual instrument for measuring the end-of-life (remaining useful life) of equipment in general. An equipment end-of-life instrument displays information regarding predicted condition of incipient failures, the uncertainty associated with them, anticipated remaining end-of-life and the uncertainty associated with it, predicted equipment efficiency, the cost associated with predicted efficiency degradation, future anticipated repairs, if any, the associated costs with them, and the cost associated with the downtime that will be needed to perform the predicted repairs. The embodiment of this instrument makes use of the algorithms in disclosure TAMUS 1059. Implementation of the instrument using other algorithms is feasible. One such implementation for an AC induction motor is given.

Fig. 1

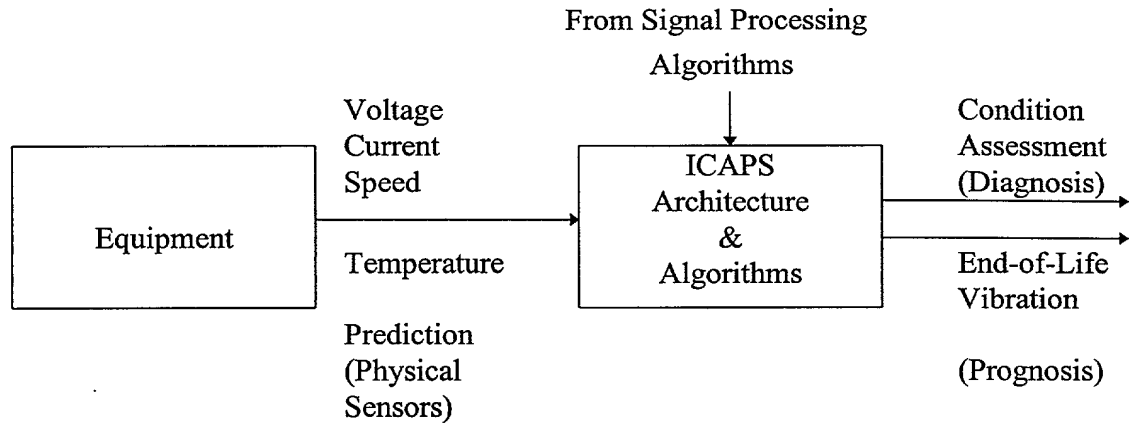


Fig. 2

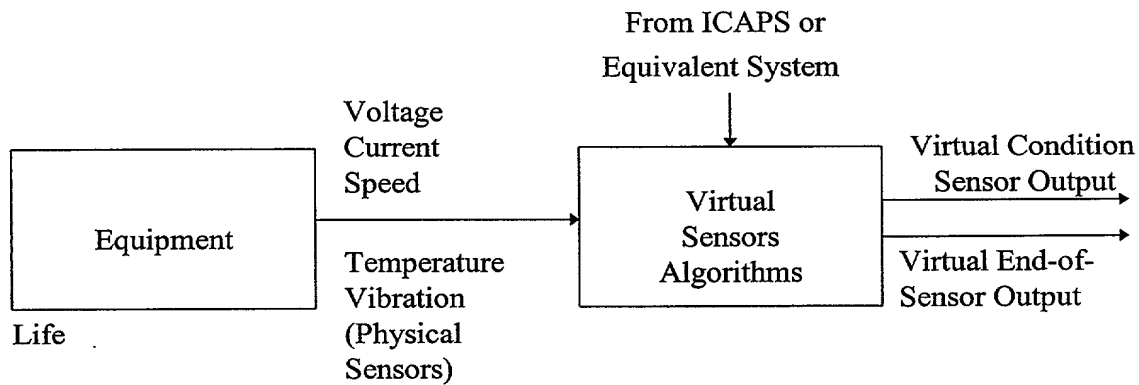


Fig. 3

DECLARATION AND POWER OF ATTORNEY

As the below-named inventor(s), I hereby declare
that:

My residence, post office address and citizenship
are as stated below next to my name.

I believe I am the original, first and sole inventor
(if plural, names are listed below) of the subject matter
which is claimed and for which a patent is sought on the
invention, design or discovery entitled SYSTEM AND METHOD FOR
CONDITION ASSESSMENT AND END-OF-LIFE PREDICTION, the
specification of which (check one):

 X is attached hereto.
 was filed on _____ as Application
Serial No. _____ and was amended
on _____ (if applicable).

I hereby state that I have reviewed and understand
the contents of the above-identified specification, including
the claims, as amended by any amendment(s) referred to above;
that I do not know and do not believe that said invention,
design or discovery was ever known or used in the United
States of America before my invention or discovery thereof, or
patented or described in any printed publication in any
country before my invention or discovery thereof, or more than
one year prior to this application, or in public use or on
sale in the United States of America more than one year prior
to this application; that said invention, design or discovery
has not been patented or made the subject of an inventor's

certificate issued prior to the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns; and that I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information known to me which is material to the patentability as defined in 37 C.F.R. § 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application(s) for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

<u>Number</u>	<u>Country</u>	<u>Date Filed</u>	<u>Priority Claimed</u>
			(Yes) (No)

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below.

<u>60/081,848</u>	<u>April 15, 1998</u>
(Serial No.)	(Filing Date)

I hereby claim the benefit under 35 U.S.C. § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application(s) in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information known to me to be material to

patentability as defined in 37 C.F.R. § 1.56 which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

<u>Application</u> <u>Serial Number</u>	<u>Date Filed</u>	<u>Status</u>
None		

I hereby appoint:

WILLIAM N. HULSEY III	Registration No. 33,402
STEPHEN E. REITER	Registration No. 31,192
GREGORY P. RAYMER	Registration No. 36,647
DAVID F. KLEINSMITH	Registration No. 40,050
BARRY N. YOUNG	Registration No. 27,774
TIMOTHY W. LOHSE	Registration No. 35,255
STANLEY H. KIM	Registration No. 40,047
DARLENE W. HAYES	Registration No. 33,899
RAMSEY R. STEWART	Registration No. 38,322
STEVEN R. SPRINKLE	Registration No. 40,825
MICHAEL A. HOFF	Registration No. 40,018

all of the firm of Gray Cary Ware & Freidenrich, as my attorneys with full power of substitution and revocation, to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith, and to file and prosecute any international patent applications filed thereon before any international authorities and to file any continuation, divisional, continuation-in-part, reissue or re-examination applications thereon.

Direct all telephone calls to:

STEVEN R. SPRINKLE
Telephone: (512) 457-7025

Send all correspondence to:

STEVEN R. SPRINKLE
GRAY CARY WARE ▲ FREIDENRICH LLP
100 Congress Avenue, Suite 1440
Austin, Texas 78701

I hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor:	Alexander G. Parlos
Inventor's signature:	_____
Date:	_____
Residence (City, County, State)	College Station, ____ County, Texas
Citizenship:	Greek
Post Office Address:	2533 Crosstimber College Station, Texas 77840

Full name of second inventor: Omar T. Rais
Inventor's signature: _____
Date: _____
Residence (City, County, State)
Citizenship: Morrocan
Post Office Address: Lotissement El Manar
Group O, No. 5
El Mank, Casablanca, Morroco

Full name of third inventor: Sunil K. Menon
Inventor's signature: _____
Date: _____
Residence (City, County, State) College Station, _____ County,
Texas
Citizenship: _____
Post Office Address: 301 Ball St. #1090
College Station, Texas 77840

Full name of fourth inventor: Amir F. Atiya
Inventor's signature: _____
Date: _____
Residence (City, County, State) Houston, _____ County, Texas
Citizenship: Egyptian
Post Office Address: 21 Shehab St., Apt. #14
Mortandesin, Cairo, Egypt
or
Cal Tech
1201 East California Blvd.
Steele Building
Pasadena, California 91125

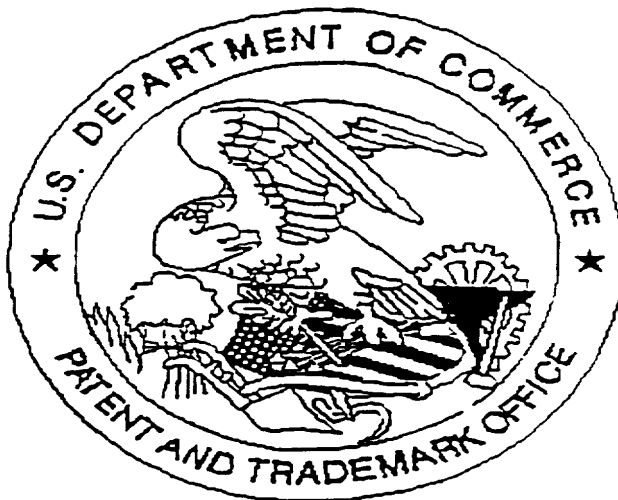
Full name of fifth inventor: Hamid A. Toliyat
Inventor's signature: _____
Date: _____
Residence (City, County, State) Bryan, _____ County, Texas
Citizenship: Iranian
Post Office Address: 1824 Wilde Oak Circle #27
Bryan, Texas 77802

Full name of sixth inventor: Esmaeil Oufi
Inventor's signature: _____
Date: _____
Residence (City, County, State) College Station, _____ County, Texas
Citizenship: Kuwait
Post Office Address: Post Office Box 2604
College Station, Texas 77841

Full name of seventh inventor: Alton D. Patton
Inventor's signature: _____
Date: _____
Residence (City, County, State) College Station, ____ County,
Texas
Citizenship: United States of America
Post Office Address: 1217 Merry Oaks
College Station, Texas 77840

Full name of eighth inventor: Benito Fernandez
Inventor's signature: _____
Date: _____
Residence (City, County, State) Austin, Travis County, Texas
Citizenship: Venezualen
Post Office Address: 3112 Thousand Oaks Drive
Austin, Texas 78746

United States Patent & Trademark Office
Office of Initial Patent Examination — Scanning Division



Application deficiencies were found during scanning:

☐ Page(s) 260-299 of Specification were not present
for scanning. 783
(Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Scanned copy is best available.